

# **THE CIP NETWORKS LIBRARY**

## **Volume 5**

### **CIP Safety**

---

Edition 2.3

November 2010



The CIP Networks Library  
Volume 5: CIP Safety Specification

Publication Number: PUB00085

Copyright © 2005-2010 ODVA, Inc. (ODVA). All rights reserved. For permissions to reproduce excerpts of this material, with appropriate attribution to the author(s), please contact ODVA at: ODVA, Inc.

4220 Varsity Drive, Suite A, Ann Arbor, MI 48108-5006 USA

TEL 1-734-975-8840

FAX 1-734-922-0027

EMAIL [odva@odva.org](mailto:odva@odva.org)

WEB [www.odva.org](http://www.odva.org)

#### Warranty Disclaimer Statement

The right to make, use, or sell product or system implementations based upon the Common Industrial Protocol (CIP) is granted only under separate license pursuant to a Terms of Usage Agreement or other agreement. The ODVA Terms of Usage Agreement is available, at standard charges, over the Internet at [www.odva.org](http://www.odva.org).

NOTE: Because the technologies described in the CIP Networks Library may be applied in many diverse situations and in conjunction with products and systems from multiple vendors, the user and those responsible for specifying these technologies must determine for themselves their suitability for the intended use. ALL INFORMATION PROVIDED BY ODVA IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, AND ODVA AND ITS MEMBERS, PARTICIPANTS, SPECIAL INTERESTS GROUPS, EXECUTIVE DIRECTOR AND BOARD OF DIRECTORS EXPRESSLY DISCLAIM ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR OR INTENDED PURPOSE, OR ANY OTHER WARRANTY OTHERWISE ARISING OUT OF THE SPECIFICATIONS. ODVA AND ITS MEMBERS, PARTICIPANTS, SPECIAL INTERESTS GROUPS, EXECUTIVE DIRECTOR AND BOARD OF DIRECTORS DO NOT WARRANT THAT USE OF THE SPECIFICATIONS (INCLUDING, WITHOUT LIMITATION, THE MANUFACTURE, DISTRIBUTION AND SALE OF PRODUCTS THAT COMPLY WITH THE SPECIFICATIONS) WILL BE ROYALTY-FREE. The user should always verify interconnection requirements to and from other equipment, and confirm installation and maintenance requirements for their specific application. IN NO EVENT SHALL ODVA, ITS OFFICERS, DIRECTORS, MEMBERS, AGENTS, LICENSORS, OR AFFILIATES BE LIABLE TO YOU, ANY CUSTOMER, OR THIRD PARTY FOR ANY DAMAGES, DIRECT OR INDIRECT, INCLUDING BUT NOT LIMITED TO LOST PROFITS, DEVELOPMENT EXPENSES, OR ANY OTHER DIRECT, INDIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES.

The following are trademarks of ODVA:

CIP CIP Motion

CIP Safety

CIP Safety CONFORMANCE TESTED

CIP Sync

DeviceNet

DeviceNet CONFORMANCE TESTED

CompoNet

CompoNet CONFORMANCE TESTED

ControlNet

ControlNet CONFORMANCE TESTED

EtherNet/IP

EtherNet/IP CONFORMANCE TESTED

All other trademarks referenced herein are property of their respective owners.



#### SITE SUBSCRIPTION

- The Final Specification, of which this volume and edition of The CIP Networks Library is a part, is provided on an annual subscription basis to this Licensed Vendor Member, as defined by its unique Vendor ID for the technology contained in the Final Specification (“YOU”), pursuant to your Terms of Usage Agreement with ODVA, Inc. (ODVA) for the technology contained in the Final Specification.
- This subscription is a site subscription, and this subscription, along with your membership in ODVA, must be renewed annually in order to maintain continuing rights to the site subscription as allowed under your Terms of Usage Agreement.
- This site subscription permits access to the electronic files for this volume contained on the distribution CD by multiple users who are your employees and on-site contracted individuals performing typical employee functions on a contract basis (“Authorized Users”). YOU shall ensure that, if access to these files is given to contractors, the contractors can fulfill the obligations of your Terms of Usage Agreement for the ODVA technology contained in the Final Specification. YOU shall not knowingly permit anyone other than Authorized Users to access these files.
- This site subscription permits access to the electronic files contained on the distribution CD for this volume via the original CD on which this volume is distributed or via an electronic copy of this volume placed on your single, secure intranet site. If and when the electronic files are posted on your intranet site, YOU shall relocate the original CD to secure storage for use only as a system back-up for the electronic files posted on your intranet site.
- The possession of or subscription to this Final Specification does not, by itself, convey any right to use or reproduce any portion of the Final Specification or to make, have made, use, import, offer to sell, sell, lease, market, or otherwise distribute or dispose of any products contemplated by the Final Specification, and you are hereby notified that the products contemplated by this Final Specification might be covered by valid patents or copyrights of ODVA, its members or other licensors. The necessary licenses to use or reproduce portions of the Final Specification for use in products, or to make, have made, use, import, offer to sell, sell, lease, market, or otherwise distribute and dispose of such products may be obtained only from ODVA through its Terms of Usage Agreement, available through the ODVA web site. This license requirement applies equally (a) to devices that completely implement the Final Specification with a network port that can be issued a Declaration of Conformity (“CIP Network Devices”), (b) to components of such CIP Network Devices to the extent they implement portions of the Final Specification, and (c) to enabling technology products, such as any CIP Network protocol stack, designed for use in CIP Network Devices to the extent they implement portions of the Final Specification. Contact ODVA for a Terms of Usage Agreement if you are not already licensed.
- Any other distribution and all other electronic copies, intranet or internet postings, and any printed copies are prohibited. Printed copies of this volume may be purchased via the order form available through the ODVA web site at [www.odva.org](http://www.odva.org).
- Notwithstanding anything to the contrary herein, if in connection with bookmarking a page of the Final Specification it is reasonably necessary for an Authorized User to possess a copy of the Final Specification on the computer on which such page is bookmarked, then you may possess such copy, provided that (i) such copy is not accessible to anyone other than the Authorized User, (ii) such copy is not retained for any longer than reasonably necessary to use the bookmarked page and in any event no longer than the term of this subscription, (iii) use of such copy is limited to the uses permitted in this site subscription and (iv) such copy is not transmitted or otherwise distributed to any other person, computer or device.



This page is intentionally left blank



## The CIP Networks Library: Volume 5

### CIP Safety

#### Table of Contents

<b>Revisions</b>	- Summary of Changes in this Edition
<b>Preface</b>	- Organization of CIP Networks Specifications - The Specification Enhancement Process
<b>Chapter 1</b>	- Introduction to CIP Safety
<b>Chapter 2</b>	- Functional Requirements
<b>Chapter 3</b>	- Communications Objects
<b>Chapter 4</b>	- CIP Object Model
<b>Chapter 5</b>	- Object Library
<b>Chapter 6</b>	- Device Profiles
<b>Chapter 7</b>	- Safety Device Configuration and Electronic Data Sheets
<b>Chapter 8</b>	- Physical Layer
<b>Chapter 9</b>	- Indicators and Middle Layers
<b>Chapter 10</b>	- Bridging and Routing
<b>Appendix A</b>	- Explicit Messaging Services
<b>Appendix B</b>	- Status Codes
<b>Appendix C</b>	- Data Management
<b>Appendix D</b>	- Engineering Units
<b>Appendix E</b>	- Safety CRCs
<b>Appendix F</b>	- Safety Test Guide



## Revisions

The CIP Networks Library Volume 5: CIP Safety Edition 2.3 contains the following changes from the previous Edition. Please see the change bars on the pages noted here for specific modifications. Note: Some of the pages within the ranges noted may not contain any changes.

Chpt-Sect	Pages	Reason for Change
		<b>EDS: Safety on a subset of ports</b>
7-2.2.2	7-27	• Modify SRS206 to include Classx keyword, add format table
7-2.2.2.1	7-27	• Insert subsection for Classx example
F—4.12	F-114	• Modify SRS206 so it agrees with edit made in 7-2.2.2
		<b>Safety Format Support in EDS</b>
7-2.2.4	7-28	• Change "Extended Format" to "Safety Format" in Conn Mgr keywords table
7-2.2.4.2.2	7-29	• Change SafetyFormat value in first paragraph to "1"
7-2.2.4.4	7-33	• Change SafetyFormat value in EDS exmple to 3
		<b>Missing NV column for DeviceNet Object</b>
5-3.1	5-8	• Change caption and add NV column to Table 5-3.1
		<b>Use of Config #1 and Config #2 in EDS Connection Manager Section</b>
5-6.2	5-53	• Changed Attrib 10 Name & Description of Attribute fields to "Target Config Data"
5-6.2.1.10 5-6.2.1.11	5-60	• Changed Config #1 to Proxy Config in two places and change Config #2 to Target Config in one place.
5-6.2.1.12 5-6.2.1.13	5-60	• Changed Config #2 to Target Config in five places
5-6.2.2.7 5-6.2.2.8	5-64, 65	• Changed Config #1 and Config #2 to Target Config in two places
7-2.3	7-30	• Changed Config #1 and Config #2 to Target Config in 4 places in Table 7-2.3
		<b>3 to 250 Byte Complemented CRC calculation clarification</b>
2-1.7.1.5 2-1.7.1.6	2-17 2-18	• Remove "XOR 0xFF" from last bullet of FRS43 and FRS369
F-3.5.1.2.3 F-3.5.1.2.4	F-30 F-31	• Change "Actual" to "Complemented" & remove "XOR 0xFF" from FRS43 & FRS369
		<b>Safety Analog Device Type</b>
6-7	6-44 thru 6-90	• Insert new Device Profile for Safety Analog Device Type

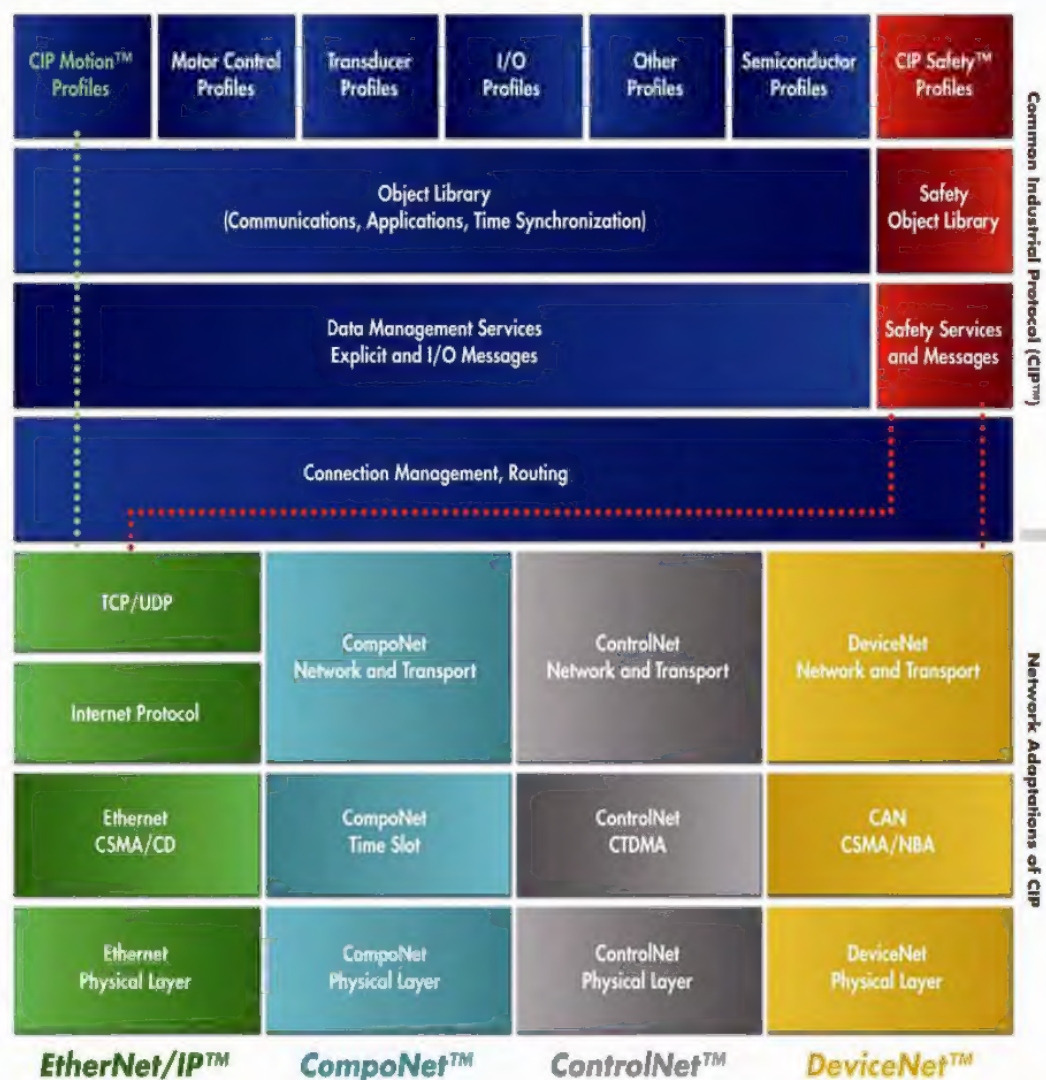


## Preface

### Organization of the CIP Networks Specifications

Today, four networks - DeviceNet™, ControlNet™, EtherNet/IP™ and CompoNet™ - use the Common Industrial Protocol (CIP) for the upper layers of their network protocol. For this reason, ODVA manages and distributes the specifications for CIP Networks in a common structure to help ensure consistency and accuracy in the management of these specifications.

The following diagram illustrates the organization of the library of CIP Network specifications. In addition to CIP Networks, CIP Safety™ consists of the extensions to CIP for functional safety.





This common structure presents CIP in one volume with a separate volume for each network adaptation of CIP. The specifications for the CIP Networks are two-volume sets, paired as shown below.

The EtherNet/IP specification consists of:

Volume 1: Common Industrial Protocol (CIP™)

Volume 2: EtherNet/IP Adaptation of CIP

The DeviceNet specification consists of:

Volume 1: Common Industrial Protocol (CIP™)

Volume 3: DeviceNet Adaptation of CIP

The ControlNet specification consists of:

Volume 1: Common Industrial Protocol (CIP™)

Volume 4: ControlNet Adaptation of CIP

The CompoNet specification consists of:

Volume 1: Common Industrial Protocol (CIP™)

Volume 6: CompoNet Adaptation of CIP

The specification for CIP Safety™ is distributed in a single volume:

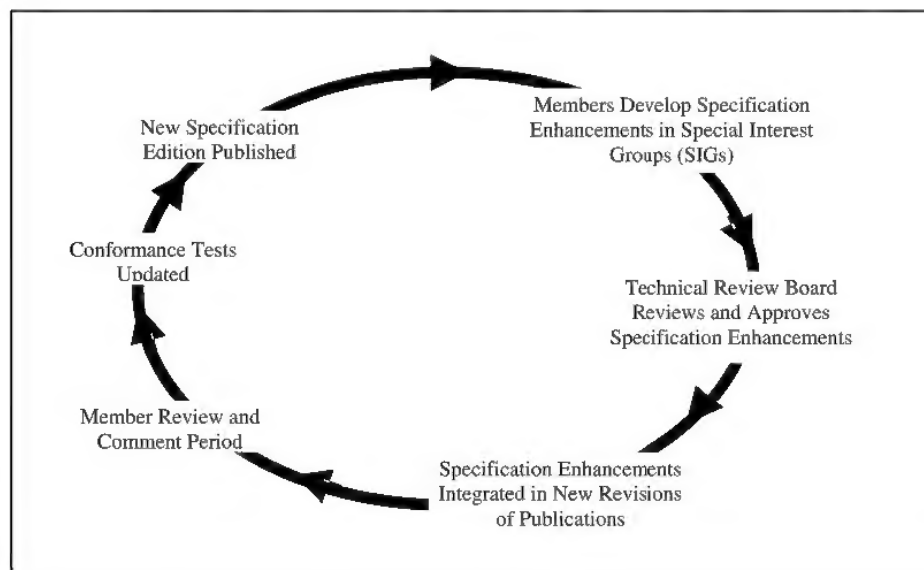
Volume 5: CIP Safety

The specification for integrating Modbus Devices is distributed in a single volume:

Volume 7: Integration of Modbus Devices into the CIP Architecture

#### Specification Enhancement Process

The specifications for CIP Networks are continually being enhanced to meet the increasing needs of users for features and functionality. ODVA has implemented a Specification Enhancement Process in order to ensure open and stable specifications for all CIP Networks. This process is ongoing throughout the year for each CIP Network Specification as shown in the figure below. New editions of each CIP Network specification are published on a periodic basis.





## **Volume 5: CIP Safety**

# **Chapter 1: Introduction to CIP Safety™**



## **Contents**

1-1	Introduction.....	3
1-2	Background.....	3
1-3	Definitions & Acronyms.....	4
1-3.1	Definitions.....	4
1-3.2	Acronyms.....	6
1-4	References.....	8



## **1-1 Introduction**

This specification identifies the functionality for a CIP safety network protocol. The scope of the design is a safety protocol used on message passing buses.

This approach uses the safety processes and coding as recommended by The German Safety Bus Committee [1] for safety data transmission on a standard network. This method relies on providing measures for possible transmission errors as defined in pr EN50159-1 [2].

## **1-2 Background**

Traditional safety systems have relied on hardwired E-stops, interlock switches, dual contact actuators and dual contact actuators connected to safety relays to provide protection for operators at run time. These systems have worked reliably but have not kept pace with developments in technology. As control systems became more complex and control systems in production lines were interconnected, the task of developing a safety system with traditional components became increasingly difficult. Complex electronic devices such as mutable light curtains and robots also made the problem more difficult to solve. Recent studies have demonstrated that run time safety is no longer the primary cause for accidents. With the realization that machine maintenance poses the largest safety risk, the difficulty in creating a safety system for complex machines is daunting when approached with traditional components.

Redundant Controllers, I/O and in some cases redundant networks have been used to solve some of these applications over the last decade. These systems were effective but provided more functionality than required and as such had limited application. For most machine applications, the requirements were very simple: shut down if a single failure is detected, prevent subsequent operation until the failure has been corrected. This requirement, known as Control Reliable is quickly gaining acceptance, in North America. In Europe first DIN 19250[4] and then the IEC 61508[5] standards are leading the way for defining the safety of electronic systems with similar requirements and will likely emerge as having worldwide acceptance.

Meanwhile a safety bus committee was formed in Germany in 1989 to define the requirements for using standard communications networks for safety applications [1]. Their initial goal was to provide a guide to apply IEC 61508 to communication networks. Their long-term goal is for this work to become part of the IEC 61508 standard. Their work is targeted primarily at high integrity (Safety Integrity Level 3) safety shutdown systems using single media on standard networks.

The approach the committee has recommended is to start with a standard, presumed unreliable, communication network and add a safety layer in the communications stack to ensure that any errors in message transmission are detected. They have based their requirements on the railway standard pr EN 50159-1[2]. This standard lists various errors that can occur and requires that at least one measure be in place to detect each of these errors. If an error is detected a safety action is taken. This means that it is possible to perform a safety function across a standard communications network by adding a safety layer that includes safety measures such as time stamps, receiver identification, CRC, time gated transmission and redundant communications. Volume 5 of CIP common defines the safety protocol using these measures and demonstrates how the common portions of this protocol can be applied to DeviceNet.

Chapter 2 presents the general case for the safety communication layer.



## Requirement Conventions

This specification uses 3 ways to identify requirements based on the type of requirement;

1. **Functional Requirement Scope** – Functional requirements are those requirements that are related to device functionality. These requirements have safety significance and are traceable to a conformance test, white-box test, or Safety Manual checklist. These requirements are identified by the numbered tag **FRSxxx**, where xxx is a unique number assigned to the requirement for traceability.
2. **System Requirement Scope** – System requirements are those requirements that are related to the safety system in which a safety device would be used. These requirements have safety significance and are traceable to a conformance test, white-box test, or Safety Manual checklist. These requirements are identified by a numbered tag **SRSxxx**, where xxx is a unique number assigned to the requirement for traceability.
3. **Interoperability requirements** – These are requirements that carry no safety significance and are not required to be traceable to a test. These requirements are identified by untaged textual statements containing the word “shall”.

## 1-3 Definitions & Acronyms

This section describes the definitions and acronyms in the context of this document.

### 1-3.1 Definitions

Term	Definition
Adapter Device	A device that is the server or target side of a network connection. These devices cannot originate any requests.
Bridge	an abstract device that connects multiple network segments along the data link layer
Connection	A persistent communications path between two devices
DeviceNet Safety	An implementation of a safety protocol on standard DeviceNet.
Failure	The termination of the ability of a functional unit to perform a required function
Dangerous Failure	A failure which has the potential to put the safety-related system in a hazardous or fail-to-function state
Hamming Distance	The number of individual bit errors in a message before an undetected error can occur in a message.
Extended Format	Abbreviated as EF. Format used for connection RPIs greater than 100ms or for applications where there is a need to avoid dropped connections.
Messaging Errors:	
Repeated message	A type of message error in which a single message is received more than once.
Lost message	A type of message error in which a message is removed from the message stream.
Inserted message	A type of message error in which an additional message is implanted in the message stream.
Re-sequenced message	A type of message error in which the order of messages in the message stream is changed.
Corrupted message	A type of message error in which a data corruption occurs.
Delayed message	A type of message error in which a message is received at a time later than intended.
Coupling of information	A type of inserted message in which a non-authentic message is designed to appear to be authentic. This may be the coupling of safety messages or the coupling of standard and safety messages.



<b>Term</b>	<b>Definition</b>
Multi-Cast	A connection between several devices where one device produces data that is consumed by one or more devices.
Network Time Expectation	The worst-case time, from a safety related event occurring as input to a Safety Producer or as a fault within the Safety Producer until the output of the Safety Consumer is put into the safety state.
Off-link Connection	A connection between two or more devices that are on physically different networks.
Base Format	Original Safety packet format for RPI's less then 100ms
Originator	A node that requests the creation of a connection
Ping	The sequence of messages that coordinates time between consumers and producers
Ping Request	Request for the acknowledge of the reception of a message.
Ping Response	Response to a ping request.
Safety Application	User related programs designed in accordance with IEC 61508 to meet the SIL requirements of the application
Safety Chain	A complete end-to-end safety function made up of sensors, inputs, control functions, outputs, and actuators.
Safety Communication Channel	A communications connection that implements the safety protocol
Safety component	A device or part of a device that is designed in accordance with IEC 61508 that meets the desired SIL level for the application
Safety Connection	A connection that utilizes the safety protocol for communications transactions.
Safety Controller	A device designed in accordance with IEC 61508 that executes a user defined safety function
Safety Data	Data that is transmitted across a safety network using a safety protocol. This data meets SIL 3 integrity level.
Safety Device	Device that contain the safety protocol.
Safety Layer	A layer of functionality in the communications stack that provides the safety protections for safety messaging
Safety Network (Bus)	An implementation of a safety protocol.
Safety Originator	A device that originates safety connections
Safety Protocol	An additional communication layer, on a standard protocol that is used to provide high integrity communications transactions.
Safety Time	The worst-case response time from physical input to physical output. This time also accounts for an error in the control system (e.g., watchdog timeout)
Safety Validator	The Safety Validator Object contains the implementation of the safety protocol on a safety device.
Safety Validator Client	a instance of a that accesses a (remote) service
Safety Validator Server	A computer software application that carries out some task on behalf of users
Scanner Device	A device that is the client or originating side of a connection. Scanner devices typically have adapter functionality as well.
Signature	A method of uniquely identifying a set of data. This could be by revision, checksum or some other method that is agreed upon by the originating and target devices.
Single-Cast	A connection between a single originator and target device.
Standard Component	Devices or portions of devices that do not participate in the safety function.
Target	A node that receives a connection creation request



### 1-3.2 Acronyms

Term	Definition
1oo2	One out of two safety architecture
BER	Bit Error Rate
BIA	Berufsgenossenschaftliches Institut für Arbeitsschutz – BIA, The BIA is an institute for research and testing of the German Berufsgenossenschaften (BG), the institutions for statutory accident insurance and prevention in Germany
CFUNID	Configuration Owning UNID
CIP	Acronym for Common Industrial Protocol. This is the application framework shared among DeviceNet, ControlNet, and Ethernet/IP.
CPCRC	Connection Parameters CRC
CP	Connection Point
CPU	Central Processing Unit
CRC	Cyclical Redundancy Check – A number derived from, and stored or transmitted with, a block of data in order to detect corruption.
CRC-Sx	Notation for use of Safety Related CRC-S1, CRC-S2 CRC-S3 or CRC-S4
DUT	Device Under Test
EDS	Device Net specific Electronic data sheet
EPATH	A data type, based upon the Abstract Syntax Notation (ASN.1), used to describe connection paths, data types, certain configuration data, etc
EPI	Expected Packet Interval
F-PLC	Failsafe programmable logical controller
EF	Extended Format. Format used for connection RPIs greater than 100ms or for applications where there is a need to avoid dropped connections
MACID	Media Access Identifier – For purposes of this specification, MACID is defined as follows: For DeviceNet safety devices, this is the node address of the device on the network. For EtherNet/IP safety devices, this is the IP address of the device on the network.
NID	Network Identifier
NTE	Network Time Expectation
NV	Non Volatile
NVS	Non Volatile Storage
OCP	Output Connection Point
OCPUNID	Output Connection Point Owning UNID
ODVA	Open DeviceNet Vendor association
OUNID	Originator Unique Network Identifier
PFD	Probability of failure on demand
PFH	Probability of dangerous failure per hour
PID	Unique Producer ID
PLC	Programmable Logical Controller
RBD	Reliability Block Diagram
SCCV	Safety Configuration Consistency Value
SCTS	Safety Configuration Time Stamp
SIL	Safety Integrity Level
SNCT	Safety Network Configuration Tool



Term	Definition
SNN	Safety Network Number
SNIL	Safety Network Interface Layer. The SNIL is the safety network protocol firmware, media dependent communication layer firmware, and required communication interface hardware (including communication processors)
SPTE	Safety Protocol Test Engine
TBD	To Be Determined
TSCCV	Targets Safety Configuration Consistency Value
TUNID	Target Unique Network Identifier
TÜV	“Technischer Überwachungsverein” – German certification and consulting organization
UCMM	Unconnected Message Manager
UNID	Unique Network Identifier



## **1-4      References**

Reference 1 - Draft proposal test and certification guideline, safety bus systems, BG Fachausschuß Elektrotechnik 28-May-2000, / GS\_ET007\_E.doc.

Reference 2 - prEN 50159-1/ August 1996: Railway applications, communication, signaling and processing systems. Part1: Safety - related communication in closed transmission systems

Reference 3 - IEC 62061 Safety of machinery - Functional Safety - Electrical, electronic and programmable electronic control systems - Draft Standard

Reference 4 - DIN V 19250/05.94: AK6 Fundamental safety aspects to be considered for measurement and control equipment

Reference 5 - IEC 61508/1999: SIL3 (superscript: Part 1-7: Functional safety of E/E/PES safety-related system)

Reference 6 - prEN 50159-2/ April 2000: Railway applications: Part 2: Safety - related communication in open transmission systems



## **Volume 5: CIP Safety**

# **Chapter 2: Safety Protocol Requirements**



## Contents

2-1	Safety Protocol Overview .....	4
2-1.1	Design Approach.....	5
2-1.2	Communication Errors and Measures to Detect Errors.....	5
2-1.3	Communication Protocol Behavior.....	8
2-1.4	Communication Layers .....	9
2-1.5	Topology .....	10
2-1.6	Supported Safety Connections .....	11
2-1.7	Safety Transmission Protocol .....	13
2-1.8	Safety Procedures.....	26
2-1.9	Safety Configuration .....	39
2-1.10	Diagnostics for Communication Errors.....	42
2-1.11	Safety Device Requirements (Device Behavior).....	43
2-2	Safety Protocol.....	44
2-2.1	High Level View of a Safety Device .....	44
2-2.2	Safety Validator Object.....	45
2-2.3	Relationship between SafetyValidatorServer and SafetyValidatorClient .....	45
2-2.4	Extended Format Time Stamp Rollover Handling.....	46
2-2.5	SafetyValidatorClient Function definition .....	49
2-2.6	SafetyValidatorServer Function Definition.....	60
2-3	CIP Connection Establishment .....	75
2-3.1	Overview .....	76
2-4	Safety Message Data Definitions .....	85
2-4.1	Mode Byte.....	85
2-4.2	Time Stamp Section .....	86
2-4.3	Time Coordination Message .....	87
2-4.4	Time Correction Message .....	87
2-4.5	Safety Data Production .....	88
2-4.6	Consumer Data Variables .....	101
2-5	Safety System Introduction .....	107
2-6	Safety Connection Establishment .....	108
2-6.1	Configuring Safety Connections .....	108
2-6.2	Establishing Connections .....	112
2-6.3	Recommendations for Consumer Number Allocation .....	115
2-6.4	Recommendations for Connection Establishment.....	115
2-6.5	Ownership Establishment.....	116
2-6.6	Ownership Use Cases.....	116
2-6.7	PID/CID Usage and Establishment.....	120
2-6.8	Network Supported Services.....	125
2-7	System Reaction Time .....	130
2-7.1	Introduction.....	130
2-7.2	Network Time Expectation .....	130
2-7.3	Produced Data Network Reaction Time.....	131
2-7.4	Equations for Calculating Network Reaction Times.....	132
2-8	Network PFH .....	134
2-8.1	Summary of PFH results for Extended .....	136
2-8.2	PFH Calculation for DeviceNet Safety Messages.....	136
2-8.3	PFH Calculation for EtherNet/IP and SERCOS III Safety Messages .....	137
2-9	DeviceNet Requirements .....	137
2-9.1	Safety Network Implications on DeviceNet.....	137
2-9.2	Safety System Throughput Over DeviceNet .....	143
2-9.3	Bus Structure.....	146
2-10	EtherNet/IP Requirements .....	148
2-10.1	EPI Rules for Safety Messages That Travel Over EtherNet/IP.....	148



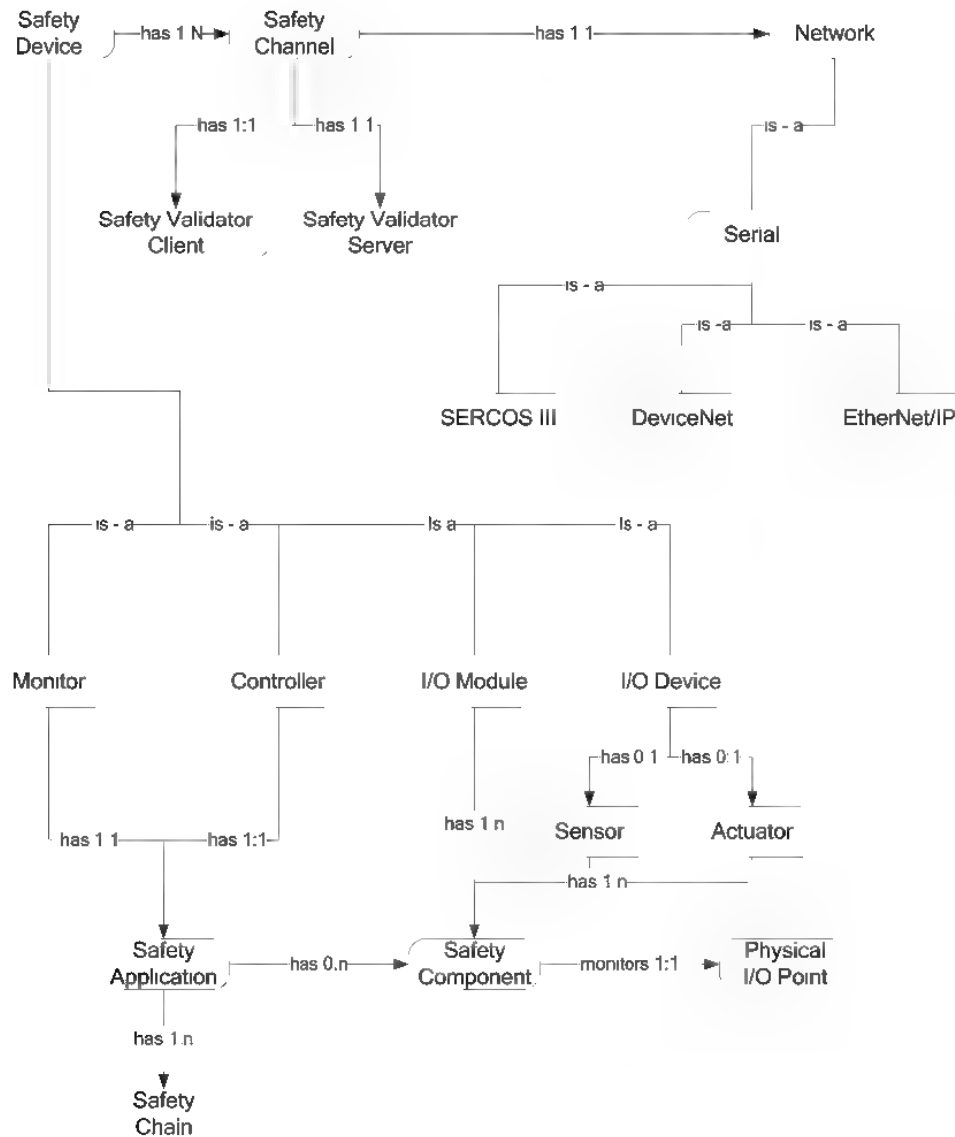
2-10.2	Default Safety I/O Service .....	148
2-10.3	Duplicate IP Detection .....	148
2-10.4	Priority for Safety Connections .....	148
2-10.5	EtherNet/IP Networks and Safety Network Numbers .....	148
2-11	SERCOS Requirements .....	150
2-11.1	Baseline CIP Safety on SERCOS Device .....	150
2-11.2	Transport Layer Requirements .....	151
2-11.3	Multicast Connections .....	151
2-11.4	CIP Safety and the SERCOS III device model .....	151
2-11.5	UNID Assignment on SERCOS III .....	152



## 2-1 Safety Protocol Overview

The safety network is designed as a protocol independent safety layer that resides above the existing network or backplane protocol. The entity-relationship shown in Figure 2-1.1 shows the relationship between possible devices, networks and components.

**Figure 2-1.1 Entity Relationship Diagram of Target Safety System**



In Figure 2-1.1, safety devices are controllers, monitors, I/O modules or I/O devices. I/O devices may be one of two types, sensor and actuator. Controllers and monitors originate communications with devices. I/O devices, I/O modules, monitors and controllers are all types of devices. Safety devices can have one or more safety communication channels, each of which can be connected to a single network. Each safety channel has a Safety Validator Client and a Safety Validator Server.



The entity relationship diagram in Figure 2-1.1 shows an entire safety application residing in a single controller or monitor. A safety application can contain multiple safety components that, in turn, can contain other safety components. A safety application will contain one or more safety chains. Much like the safety application, the safety chain can contain multiple safety components that, in turn, can contain other safety components. The resulting structure for a safety application is, therefore, a tree of safety components, with each having access the I/O points within the safety system.

## 2-1.1 Design Approach

The CIP Safety approach uses the safety processes and coding recommendations of the German Safety Bus committee, as documented in the German Safety Bus Committee Specification, Appendix A. that relies on providing measures for possible transmission errors as initially defined in ISO-62280-1. This specification requires a single measure to detect each error condition. The CIP Safety approach exceeds these basic requirements and provides alternate detection measures where possible. Table 2-1.1, is based on the Error and Measures table documented by the German Safety Bus Committee. Table 2-1.1 shows just those measures that the safety protocol uses to detect errors. Also provided in the table are references within this specification where the detection measures are fully described; thus, this table can be used as the central starting point in analyzing the safety protocol.

## 2-1.2 Communication Errors and Measures to Detect Errors

Table 2-1.1 Measures Against Errors in Messages

Communication Errors	Measures to detect communications errors				
	Time Expectation via time stamp	Identification for sender and receiver	CRC	Redundancy with Cross Checking	Different data integrity assurance systems for safety and standard messages
Message Repetition 2-1.2.1	X		X <sup>2</sup>		
Message Loss 2-1.2.2	X		X <sup>2</sup>		
Message Insertion 2-1.2.3	X	X	X <sup>2</sup>		
Incorrect Sequence 2-1.2.4	X		X <sup>2</sup>		
Message Corruption 2-1.2.5			X	X	
Message Delay 2-1.2.6	X				
Coupling of safety and safety information 2-1.2.7		X			
Coupling of safety and standard information 2-1.2.8	X	X	X	X	X
Increased age of data in bridge <sup>1</sup> 2-1.2.9	X				

1. This requirement is not part of the German Safety Bus Committee Specification

2. The Safety CRC provides additional protection for communication errors in fragmented messages.



### **2-1.2.1 Message Repetition**

It is possible for a message to be repeated on a network. This in itself does not represent an error, because the safety protocol allows overwriting of data. However, a repeated message will not have an updated time stamp, because only the base producer can update the time stamp. Repeated messages that were received in place of new data may result in the connection's termination (2-1.3.2) if they did not meet the consumer's time expectation.

### **2-1.2.2 Message Loss**

Time Expectation detects loss of a message. FRS3 The safety protocol requires messages to occur within defined time expectations. Messages received later than these time expectations shall be treated as errors, resulting in the connection's termination.

### **2-1.2.3 Message Insertion**

Message insertion is detected by two measures:

Time Expectation: An inserted message will result in a connection termination because the message received will have an unexpected value in its time-stamp or time-stamp/roll-over count.

An inserted message is detected by identification of sender and receiver. A unique identifier is encoded with the CRC-Sx of the time stamp section, time coordination section, and both of the data sections (2-1.7.1). If the CRC-Sx is incorrect, either the data is corrupted or a message has been sent to the wrong device. See FRS5 (Section 2-1.2.5) to see how this error is handled..

### **2-1.2.4 Incorrect Sequence**

The time expectation detects an incorrect sequence. The incorrect sequence of a message will result in a connection termination because the message received has an unexpected value in its time-stamp and/or roll-over count.

### **2-1.2.5 Message Corruption**

Two additional measures of safety networks exceed the native measures of standard networks to detect message corruption:

Cyclic Redundancy Check: A safety cyclic redundancy code, CRC-Sx (2-1.7.2), is encoded in each safety message.

FRS5 A message corruption shall be detected when the data and CRC-Sx are calculated and compared. This error shall cause the connection to be

Terminated

IF (Base format) OR (Extended Format AND Consumer\_Fault\_Count) >= Max\_Fault\_Number  
OR Dropped

IF Extended Format AND Consumer\_Fault\_Count < Max\_Fault\_Number.



Redundancy with cross checking: All safety data is sent twice (one copy is the ones complement) in the same message packet. The received safety data is crosschecked when it arrives at the consumer. FRS6 A corrupted message that was not detected by the link or CRC-Sx check (i.e., error that exceeded the Hamming distance of CRC) shall be detected when the actual and complemented copies of the data are compared in the consumer. See Section 2-1.8 for details.

#### **2-1.2.6 Message Delay**

Time Expectation detects message delay. If an expected message is not received at the consumer during the required time interval, the connection's periodic timer will expire and the connection shall be terminated (2-1.3.2) See Section 2-7 for a detailed description of safety network times.

#### **2-1.2.7 Coupling of Safety and Safety Information**

The coupling of safety messages is detected by the inclusion of a unique identifier, called the PID in the time stamp section, data sections and time correction message Safety CRC calculations. It is called the CID in the time coordination message Safety CRC calculations (2-1.7.1). The PID and CID are established at connection time and is not transmitted with the data. FRS7 The PID or CID is incorporated within the Safety CRC calculation in both the producer and the consumer, thus, if a message is mistakenly received, it shall be detected when the CRC is checked.

#### **2-1.2.8 Coupling of Safety and Standard Information**

All five of the safety detection measures are designed to detect the coupling of safety and standard information because a standard message will not use the safety format. FRS8 Safety consumer shall detect standard messages as an incorrectly formed message.

1. Redundancy with cross check: Any standard message that was inadvertently received by a safety consumer will not meet the requirement for redundant data in the same link packet and therefore an error will occur when cross checking is attempted, See Section 2-1.8.
2. Different data integrity assurance systems for safety and standard devices: Safety messages have a unique message encoding (2-1.7), which includes a time stamp and a unique Safety CRC-Sx. A standard device connection will not be able to encode information in this format or generate the correct cyclic redundancy code of a safety message.
3. A standard message will not contain the Safety CRC (as generated by the Safety CRC algorithm) and will not construct the message using the correct safety format, thus a standard message will be detected as an error condition.
4. A standard message will not contain a correct timestamp; this shall cause an error to be detected.
5. A standard message will not contain the correct packet format; this shall cause an error to be detected.

#### **2-1.2.9 Increased Age of Data in Bridges**

A time stamp using time expectation is used to detect possible increased age of data in bridges. If an expected message is not received at the consumer by the required time interval, the connection will be terminated (2-1.3.2). See Section (2-1.8.1) for a detail explanation of the time stamp protocol.



### **2-1.2.10 Addressing errors**

This section describes the measures used to detect addressing errors.

#### **2-1.2.10.1 Producer generates an incorrect address, changes header**

1. A message is somehow misdirected to an invalid address. This condition detected by: Message Loss (2-1.2.2) in the original consumer.
2. A Message is somehow misdirected to another safety device, This condition is detected by: Message Insertion (2-1.2.3), Repetition (2-1.2.1), Incorrect Time Stamp (2-1.2.4), Message Delay (2-1.2.6), and, Coupling of Safety and Safety Information (2-1.2.7).

#### **2-1.2.10.2 Consumer consumes a wrong address**

1. A safety device consumes a message intended for another non-safety related device. This condition is detected by Coupling of Safety and Standard Information (2-1.2.8).
2. A safety device consumes a message intended for another safety device. This condition is detected by: Message Insertion (2-1.2.3), Message Repetition (2-1.2.1), Identification for receiver, and Incorrect Time Stamp (2-1.2.4), and, Coupling of Safety and Safety Information (2-1.2.7).

### **2-1.2.11 Measures to Protect Standard Devices from Safety Devices**

Safety devices are built on the standard network protocol and do not impact standard devices other than using available bandwidth. Improper allocation of bandwidth may cause nuisance trips to the safety system, causing it to go to a safety state. See section (2-1.8.1).

## **2-1.3 Communication Protocol Behavior**

### **2-1.3.1 Sequence of Safety Checks**

FRS9 The following Sequence of checks shall be used to check messages:

- Ping count check
- Evaluate Time stamp section CRC
- Evaluated Time Stamps
- Evaluate Actual Data CRC
- Evaluate Complement Data CRC
- Perform Cross-Check

### **2-1.3.2 Connection Termination**

FRS10 When a producer detects an error that requires connection termination it shall terminate the connection, and notify the application of the action.

FRS11 All consumers shall monitor the periodic transmission of data and go to a safety state if the periodic transmissions cease. If the consumer detects an error, it must go to a safety state and terminate the consuming connection associated with that error.

FRS12 When the Safety Layer detects an error requiring connection termination the termination shall be implemented using the following sequence.



- The safety layer detects an error requiring termination
- The safety layer notifies the application program by setting the connection status to indicate a safety communications fault
- The safety application shall transition data and I/O (e.g. set outputs to the safety state) associated with the connection to a safety state
- The safety layer shall notify the underlying communications system of an error and request the termination of the connection by setting its status to indicate a safety communications error
- The safety layer shall not transition from its safety state until the fault is cleared and a connection restart sequence is initiated.

### **2-1.3.3 Cross Checking Error**

FRS13 When crosschecked safety data are found to be different, the data is treated as faulty. The base format consumer shall terminate the connection (See Section 2-1.3.2) and the Extended Format consumer will increment the Consumer\_Fault\_Count and drop the packet if the count is less than the Max\_Fault\_Number, or terminate the connection if is equal to or greater than the Max\_Fault\_Number.

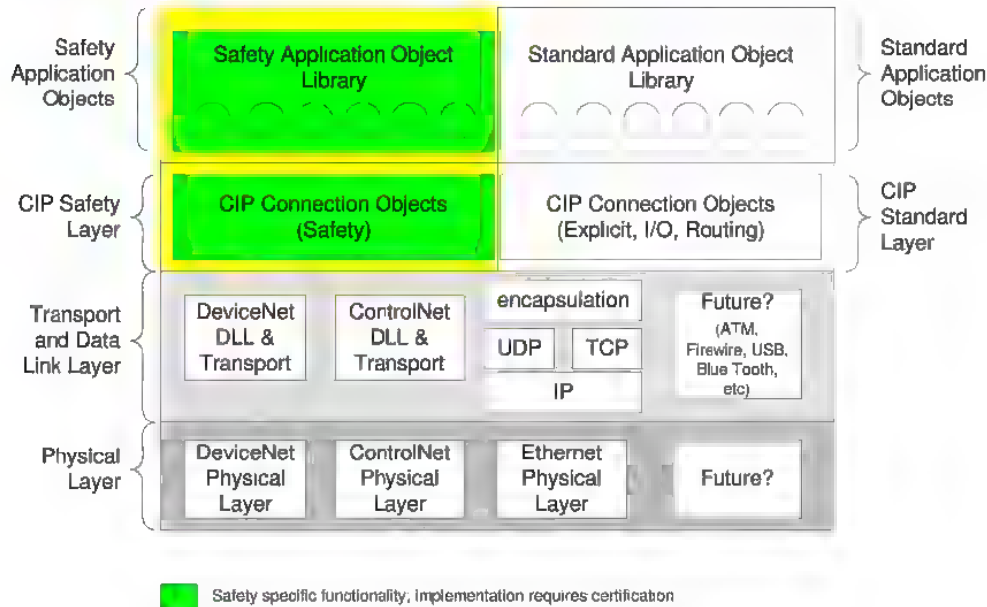
### **2-1.4 Communication Layers**

The safety protocol is layered on top of the standard network protocol.

The Safety Layers accept safety data from the safety application circuit. The safety layers then formulate a safety message, transmit the safety message, receive and decode the safety message. Once the safety message is decoded and verified, the data is presented to the receiving safety application circuit



Figure 2-1.2 Communication Layers



## 2-1.5 Topology

### 2-1.5.1 Logical

The logical information flow is identical for both systems, as shown in Figure 2-1.3. This flow consists of the gathering of safety information at the safety input, then a safety transmission of this information to the controller. The safety information is processed and the resulting safety information is transmitted to the safety output, where the safety action takes place.

Figure 2-1.3 Logical information flow<sup>1</sup>



#### 2-1.5.1.1 Addresses for Safety Devices

FRS14 Each safety device shall have a single physical address that is unique on the device's network segment.

<sup>1</sup> Note: All safety subsystems shall meet the requirements for the desired Safety Integrity Level (SIL) for the application



### 2-1.5.2 Physical Topologies

FRS15 The safety network shall not restrict any physical topologies.

### 2-1.6 Supported Safety Connections

There are two types of safety connections defined for the safety protocol:

- Single-Cast
- Multi-Cast

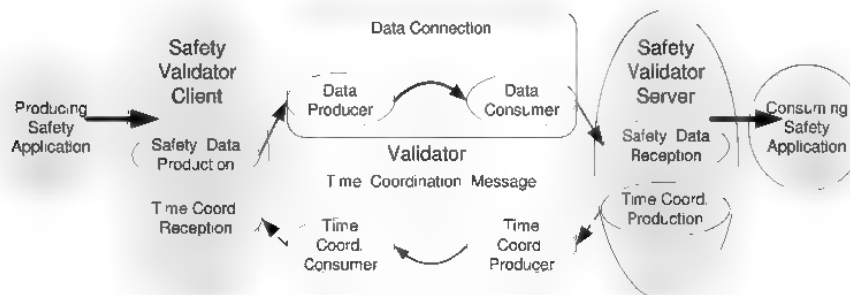
#### 2-1.6.1 Single-Cast Safety Connections

FRS16 For single-cast safety connections, the Single-cast SafetyValidatorClient shall produce safety data to the Single-cast SafetyValidatorServer as defined by the Expected Packet Interval (EPI) rate.

FRS17 The Single-cast SafetyValidatorServer shall respond to a ping request and produces an Time Coordination message with a Time\_Value to the Single-cast SafetyValidatorClient that is used by the safety data producer.

FRS18 The Single-cast safety data producer shall use the time value returned in the ping response (Time Coordination message) to adjust the time stamp sent the Single-cast produced data. The time stamp is relative to the safety data consumer's clock.

Figure 2-1.4 Single-Cast Block Diagram



In the diagram above, the safety critical processing is performed within the shaded areas. A single-cast safety connection will require two transport connections.

#### 2-1.6.2 Multi-Cast Safety Connections

FRS23 For multi-cast safety connections the SafetyValidatorClient shall produce Safety Data to multiple ( $n = 2$  to 15) SafetyValidatorServers as defined by the Expected Packet Interval (EPI) rate.

FRS24 The produced Multi-Cast safety data shall include a Producer\_Time\_Stamp with each safety data production.

FRS25 The Multi-Cast SafetyValidatorServers shall respond to a ping request by sending an Time Coordination message with a Consumer\_Time\_Value to the SafetyValidatorClient.



FRS26 The Multi-cast Consumer\_Time\_Values shall be used by the Multi-cast safety data producer to generate a Time\_Correction\_Value for each safety data consumer .

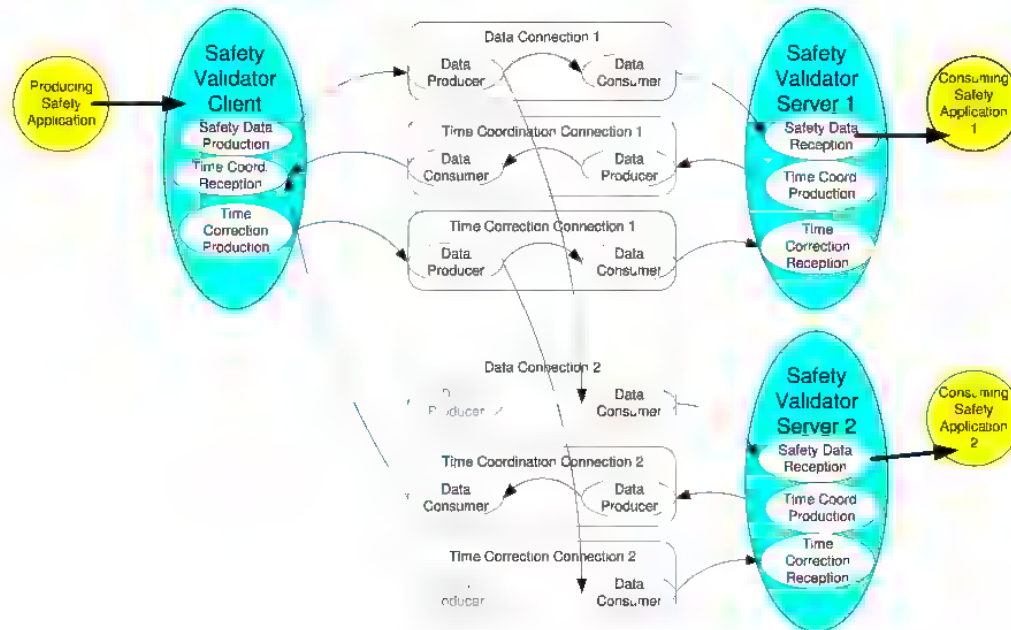
FRS27 A Time\_Correction\_Value shall be sent by the Multi-cast safety data producer to each safety data consumers each Ping Interval.

FRS28 The Multi-cast safety data consumer shall use the Producer\_Time\_Stamp and the Time\_Correction\_Value to derive a Data\_Time\_Stamp that is relative to the safety data consumer's clock.

The Time\_Correction\_Value is sent to each consumer in one of 2 ways.

FRS29 For devices on a DeviceNet network, the Multi-cast Time\_Correction\_Value shall be sent to the consumers via a separate multi-cast trasport connection.

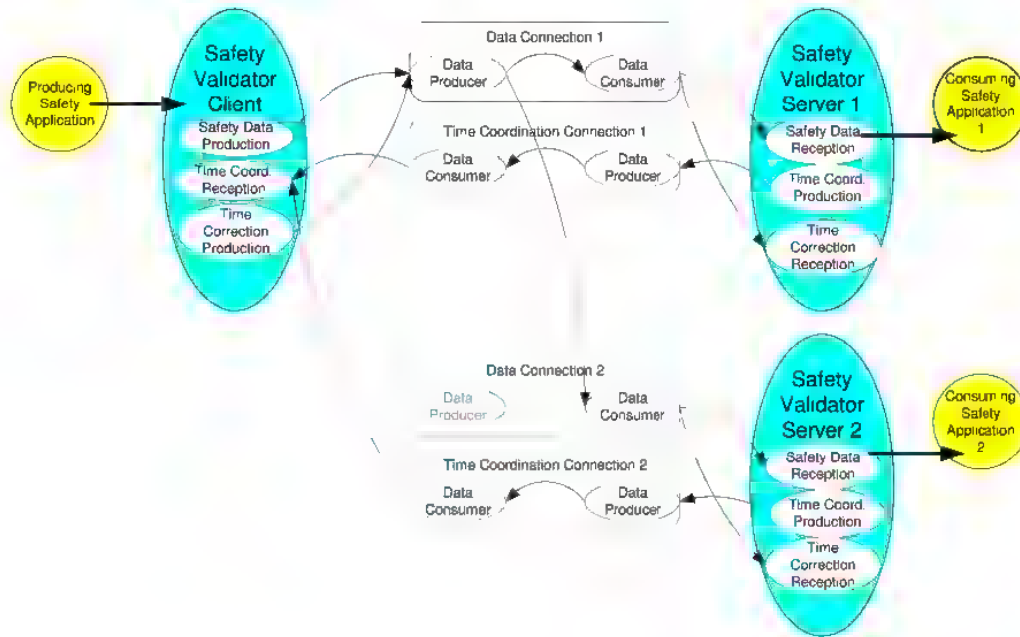
**Figure 2-1.5 Multi-Cast Block Diagram - DeviceNet**



FRS30 For devices on a non-DeviceNet network, the data and Time\_Correction\_Value shall be concatenated and sent to the consumers via a single multi-cast transport connection.



Figure 2-1.6 Multi-Cast Block Diagram, non-DeviceNet



In the diagram above the safety critical processing is done within the shaded areas. A multi-cast safety connection shall require 3 times  $n$  transport connections for DeviceNet devices, and 2 times  $n$  transport connections for non-DeviceNet devices..

## 2-1.7 Safety Transmission Protocol

FRS32 All safety messages and safety response messages shall be encoded with the safety CRC's defined in Appendix E .

### 2-1.7.1 Safety Message Encoding

The Safety Protocol defines nine section encoding formats that are used in different combinations for safety messaging.

- Base format 1 or 2 byte Data Section (See section 2-1.7.1.2)
- Extended format 1 or 2 byte Date Section (See section 2-1.7.1.4)
- Base format 3 to 250 byte Data Section (See section 2-1.7.1.5)
- Extended format 3 to 250 byte Data Section (See section 2-1.7.1.6)
- Base format Time Stamp Section (See section 2-1.7.1.7)
- Base format Time Coordination Section (See section 2-1.7.1.8)
- Extended format Time Coordination Section (See section 2-1.7.1.8)
- Base format Time Correction Section (See section 2-1.7.1.8.2)
- Extended format Time Correction Section (See section 2-1.7.1.8.2)

These section encodings are used to construct six safety message types. In all six types, they can use either the Base format or the Extended format.



The six types of safety message are:

- 1 or 2 byte Single-Cast (See section 2-1.7.1.9)
- 1 or 2 byte Multi-Cast DeviceNet (See section 2-1.7.1.10)
- 1 or 2 byte Multi-Cast Non-DeviceNet (See section 2-1.7.1.11)
- 3 to 250 byte Single-Cast (See section 2-1.7.1.12)
- 3 to 250 byte Multi-Cast DeviceNet (See section 2-1.7.1.13)
- 3 to 248 byte Multi-Cast Non-DeviceNet (See section 2-1.7.1.14)

The following table shows which Sections are used within the various message types:

**Table 2-1.2 Connection Sections and Message Types**

Safety Connection Format			Data Message <sup>2</sup>				Time Coordination Message <sup>2</sup>	Time Correction Message <sup>2</sup>
Data Size	Safety Connection Type	Network Hop Type	1 or 2 byte data section	3 to 250 byte data section	Time Stamp section <sup>1</sup>	Time Correction Section	Time Coordination Section	Time Correction Section
1 or 2 bytes	Single-Cast	All	X		X <sup>1</sup>		X	
	Multi-cast	DeviceNet	X		X <sup>1</sup>		X	X
		Non-DeviceNet	X		X <sup>1</sup>	X	X	
3 to 250 Bytes	Single-cast	All		X	X <sup>1</sup>		X	
3 to 248 Bytes	Multi-cast	DeviceNet		X	X <sup>1</sup>		X	X
	Multi-cast	Non-DeviceNet		X	X <sup>1</sup>	X	X	

1 Base format has independent section for Time Stamp; it is part of the data section in the Extended format

2 Message can use either Base or Extended format.

Originators and Targets shall follow selection criteria shown in Table 2-1.3 for the usage of the Base and Extended Formats.

**Table 2-1.3 Connection Sections and Message Formats**

Type	Base Format	Extended Format
Originator	Required <sup>1</sup>	Required
Target	Required <sup>1</sup>	Required

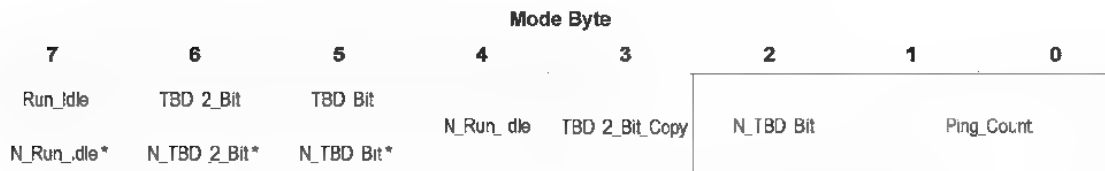
<sup>1</sup> May be optional in future editions



### 2-1.7.1.1 Mode Byte

The Mode Byte is described outside of the section definitions because various parts are included in the various section CRCs. In the base format, bits 0 thru 4 of the mode byte are part of the time stamp section, while bits 5 thru 7 are part of the data sections. In the short packet Extended Format, bits 5 thru 7 are part of the data CRC. In the large packet Extended Format bits 0 thru 4 of the mode byte are part of the complement data CRC, while bits 5 thru 7 are part of the true data CRC. The variables used in the mode byte, brief descriptions and references to complete descriptions are given in Table 2-1.4.

Figure 2-1.7 Format of the Mode Byte



\* These values are not sent in the data packet, they are derived by the consumer

Table 2-1.4 Mode Byte Variables

Name	Description	Reference
Run_Idle	FRS34 Run_Idle shall be used to indicated the usability of the data as determined by the Producer Safety Application	2-4.1.1
N_Run_Idle	Complement of the Run_Idle	2-4.1.2
TBD_2_Bit	Reserved for future use	2-4.1.3
TBD_2_Bit Copy	Copy of the TBD_2_Bit	2-4.1.4
TBD_Bit	Reserved for future use FRS126 The TBD_Bit Shall be set to 0 by the Safety Validator	2-4.1.6
N_TBD_Bit	Complement of the TBD_Bit	2-4.1.7
Ping_Count	The ping count is used for determining when a consumer should send a time coordination message	2-4.1.5

### 2-1.7.1.1.1 Mode Byte CRC Processing for Base Format

FRS37 The Mode Byte processing within the Base Format CRCs shall be done as shown as follows.

- CRC Type Mode Byte logic prior to inclusion within the CRC
- Actual Data CRC                      Mode Byte **AND** 0xE0
- Complement Data CRC              (Mode Byte **XOR** 0xFF) **AND** 0xE0
- Time Stamp Section CRC            Mode Byte **AND** 0x1F

### 2-1.7.1.2 1 or 2 Byte Data section, Base Format

FRS38 The 1 or 2 Byte Data section for the base format shall consist of the actual data byte or bytes, the mode byte, the actual CRC, and the complement CRC.



Figure 2-1.8 1 or 2 Byte Data Section Base Format

1 or 2 Byte Data Section, Base Format																																							
Actual Data																Mode Byte								Actual CRC								Comp. CRC							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0								
1 – 2 Bytes																						CRC-S1						CRC-S2											

FRS39 The Base format 1 or 2 Byte Actual Data CRC calculation shall include:

- Producer Identifier (PID) (by CRC-S1)
- The (Mode Byte) AND (0xE0) (by CRC-S1)
- Actual Data byte(s) (by CRC-S1) (see yellow colored sections)

FRS40 The Base Format 1 or 2 Byte Complement Data CRC calculation shall include:

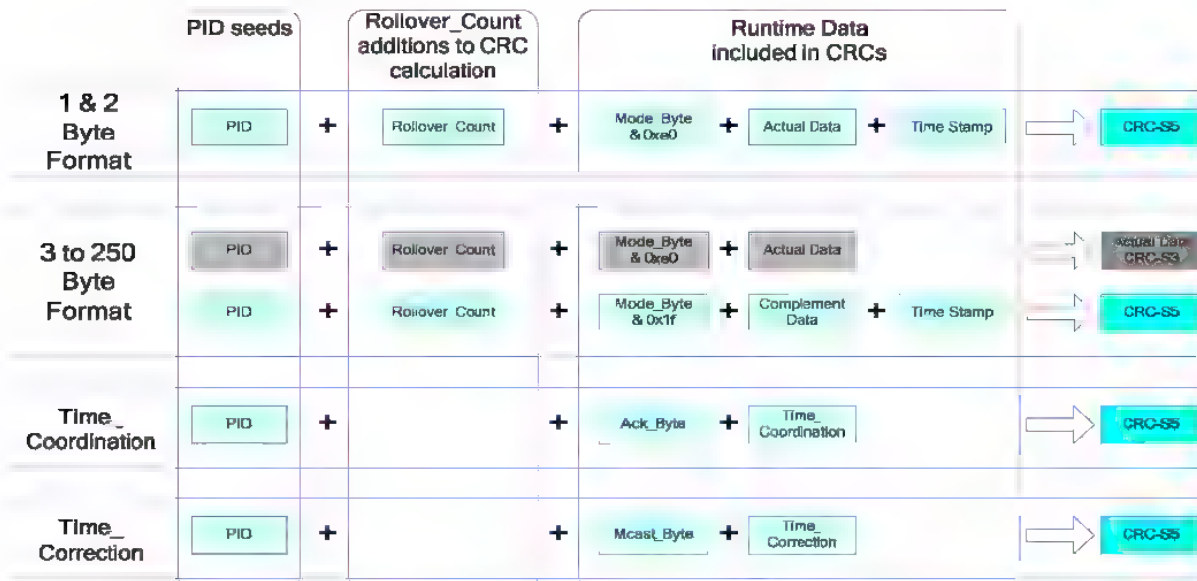
- Producer Identifier (PID) (by CRC-S1)
- The (Mode Byte XOR 0xFF) AND (0xE0) (by CRC-S2)
- Actual Data bytes XOR 0xFF (by CRC-S2) (see green colored sections)

Note that the PID uses CRC-S1 for both the actual and the complement CRC calculation. Refer to Appendix E for code samples.

### 2-1.7.1.3 Calculation order for Extended Format CRC calculations

Figure 2-1.9 below shows where the rollover count fits into the CRC calculations for the Extended Format messages. Note the CRC seed is still based on the PID/CID as in the base format; the Rollover count is treated as an additional data parameter.

Figure 2-1.9 CRC Calculation order for Extended Format messages





#### 2-1.7.1.4 1 or 2 Byte Data Section, Extended Format

FRS365 The 1 or 2 Byte Data section for the Extended format shall consist of the actual data byte or bytes, the mode byte, the time stamp, and CRC-S5 calculated over the entire packet.

Figure 2-1.10 1 or 2 Byte Data Section, Extended Format

1 - 2 Byte Data Section, Extended Format					
Actual Data	Mode Byte <sup>1</sup>	CRC S5		Time Stamp	CRC S5
2 bytes	1 byte	2 bytes		2 bytes	1 byte
1 – 2 Bytes		CRC-S5_0	CRC-S5_1	Time Stamp	CRC-S5_2

<sup>1</sup> Bits 4-0 are not included in the CRC calculation.

FRS366 The Extended 1 or 2 Byte Data CRC calculation shall use CRC-S5 and include:

- Producer Identifier (PID)
- 16-bit Rollover count
- Mode Byte & 0xE0
- Actual Data byte(s)
- Time Stamp (see yellow colored sections)

Refer to Appendix E for code samples.

#### 2-1.7.1.5 3 to 250 Byte Data section, Base Format

FRS41 The Base format 3 to 250 Byte Data section shall consist of the actual data bytes, the mode byte, the actual CRC, the complemented data bytes, and the complement CRC.

Figure 2-1.11 3 to 250 Byte Data Section, Base Format

3 - 250 Byte Data Section, Base Format				
Actual Data	Mode Byte	Actual CRC	Complemented Data	Comp. CRC
3- 250 bytes	1 byte	2 bytes	3- 250 bytes	2 bytes
3 – 250 Bytes		CRC-S3	3-250 Bytes	CRC-S3

FRS42 The Base format 3-to250 Byte Actual Data CRC calculation shall include:

- Producer Identifier (PID) (by CRC-S3)
- The (Mode Byte) AND (0xE0) (by CRC-S3)
- Actual Data byte(s) (by CRC-S3) (see yellow colored sections)

FRS43 The Base format 3-to250 Byte Complement Data CRC calculation shall include:

- Producer Identifier (PID) (by CRC-S3)
- The (Mode Byte XOR 0xFF) AND (0xE0) (by CRC-S3)
- Complemented Data bytes (by CRC-S3) (see green colored sections)

Refer to Appendix E for code samples.



### 2-1.7.1.6 3 to 250 Byte Data section, Extended Format

FRS367 The Extended 3 to 250 Byte Data section shall consist of the actual data bytes, the mode byte, CRC-S3, the complement data bytes, the time stamp and CRC-S5 .

Figure 2-1.12 3 to 250 Byte Data Section, Extended Format

3-250 Byte Extended Format							
Actual Data	Mode Byte	Actual Data CRC	Complement Data	Complement Data CRC		Time Stamp	Compl. Data CRC
3-250 bytes	1 byte	2 bytes	3 – 250 bytes	2 bytes		2 bytes	1 byte
3 – 250 Bytes		CRC S3	Complement 3-250 Bytes	CRC S5_0	CRC S5_1	Time Stamp	CRC S5_2

FRS368 The Extended format 3-to250 Byte Actual Data CRC calculation shall include:

- Producer Identifier (PID) (by CRC-S3)
- Rollover Count (by CRC-S3)
- The (Mode Byte) AND (0xE0) (by CRC-S3)
- Actual Data byte(s) (by CRC-S3) (see yellow colored sections)

FRS369 The Extended format 3-to250 Byte Complement Data CRC calculation shall include:

- Producer Identifier (PID) (by CRC-S5)
- Rollover Count (by CRC-S5)
- Mode Byte AND 0x1F (by CRC-S5)
- Complement Data bytes (by CRC-S5)
- Time Stamp (by CRC-S5) (see green colored sections)

Refer to Appendix E for code samples.

### 2-1.7.1.7 Time Stamp Section, Base Format

The Base format Time Stamp section is composed of bits 0-4 of the mode byte, the Time\_Stamp, and the time stamp CRC.

Figure 2-1.13 - Base Format, Time Stamp Section Format

Time Stamp Section																								
7	6	5	4	3	2	1	0		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
								Data	Time Stamp										CRC-S1					
								Section																
Mode Byte																								

Table 2-1.5 Time Stamp Variables

Name	Description	Reference
Data_Time_Stamp	Single-Cast -- Data_Time_Stamp is the time the data was sampled for production relative to the consumers clock Multi-Cast -- Data_Time_Stamp is the time the data was sampled for production relative to the producers clock	2-4.2.1



FRS45 The Base format Time Stamp CRC calculation shall include:

- Producer Identifier byte (PID) (by CRC-S1)
- The (Mode Byte) AND (0x1F) (by CRC-S1)
- Time Stamp byte(s) (by CRC-S1) (see blue colored sections)

### 2-1.7.1.8 Time Coordination Section

FRS48 The time coordination section shall be sent from the consumer to the producer in response to a change in the ping count.

The Time Coordination section is used as the Time Coordination Message for all single-cast and multi-cast time coordination requests. There are two formats: the base format and Extended format.

The encoding for the time coordination section is shown in Figure 2-1.14. The variables used in the time coordination section, brief descriptions and references to complete descriptions are given in Table 2-1.6.

Figure 2-1.14 Base Format Time Coordination Message Encoding

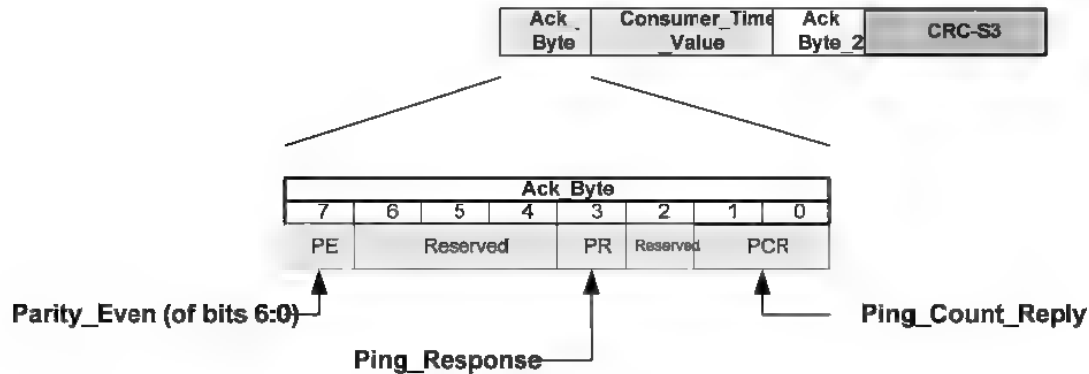


Figure 2-1.15 Extended Format Time Coordination Message Encoding

Ack_Byte	Consumer_Time_Value	CRC_S5_0	CRC_S5_1	CRC_S5_2
----------	---------------------	----------	----------	----------

Table 2-1.6 Time Coordination Message Variables

Name	Description	Reference
Consumer Time Value	FRS51 Consumer Time Value in the Time Coordination message shall (as closely as possible) mark the time at which the consumer sends a ping response message	2-4.3.2
Ack_Byte	Consisting of:	
Parity_Even	FRS52 The Parity_Even bit in the Time Coordination message shall be even parity of bits 0 through 6 of the message	
Ping_Response_Bit	FRS53 The Ping_Response_Bit in the Time Coordination message shall indicates that a ping response is being sent	2-4.3.1



Name	Description	Reference
Ping_Count_Reply	FRS55 Ping_Count_Reply in the Time Coordination message shall indicate which ping request the consumer is responding to	2-4.3.3
Reserved	Reserved, set to zero	2-4.3.4
CRC-S3	CRC covering the data within the Base Format time coordination message	
CRC-S5	CRC covering the data within the Extended Format time coordination message	

FRS56 The following rule shall be used for the calculation of the parity bit in the Time Coordination message:

Bit 7 of the Ack\_Byte form even parity on the Ack\_Byte.

FRS57 The following rule shall be used for setting of the bits in Ack\_Byte\_2 of the Time Coordination message:

$Ack\_Byte\_2 = ((Ack\_Byte \text{ XOR } 0xFF) \text{ AND } 0x55) \text{ OR } (Ack\_Byte \text{ AND } 0xAA)$

FRS370 When a connection is established with the Extended format, the EF Time Coordination format shall be used; otherwise the base format shall be used.

#### **2-1.7.1.8.1 Time Coordination CRC Calculation**

FRS58 The Time Coordination CRC calculation shall include:

- Consumer Identifier (CID) (by CRC-S3 or CRC-S5)
- Ack Byte (by CRC-S3 or CRC-S5)
- Consumer\_Time\_Value (by CRC-S3 or CRC-S5) (see *light grey colored sections*)

FRS59 Ack\_Byte\_2 in the Base Time Coordination message shall not be included in the Safety CRC .

#### **2-1.7.1.8.2 Time Correction Section**

FRS60 The time correction message shall be sent from a multi-cast producer to each multi-cast consumer, once every ping interval. The time correction message, which contains consumer specific information, is sent by the producer to the consumer at a slower rate than the produced data. The encoding for the base time correction message is in Figure 2-1.16 and the encoding for the Extended format is in Figure 2-1.17.



Figure 2-1.16 Base Format Time Correction Message Encoding

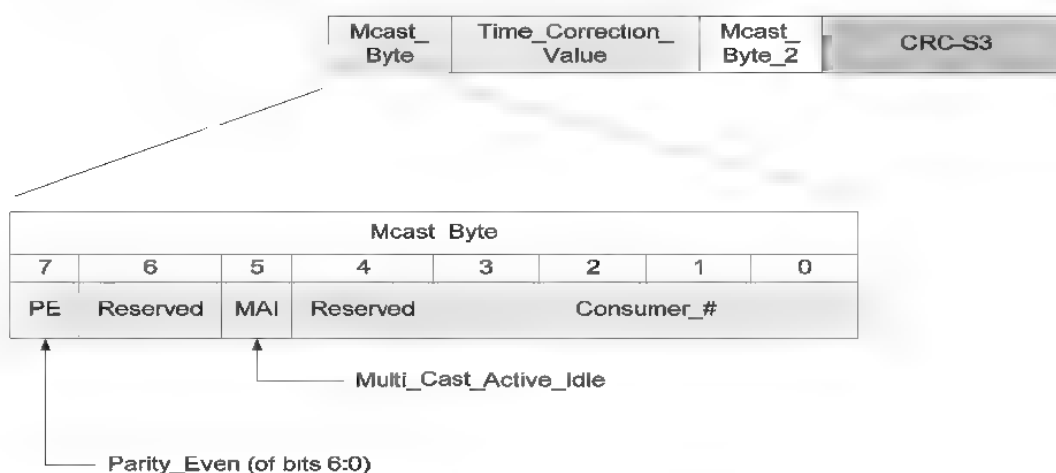


Figure 2-1.17 Extended Format Time Correction Message Encoding



Table 2-1.7 Time Correction Message Variables

Name	Description	Reference
Consumer_Time_Correction_Value	FRS61 Consumer_Time_Correction_Value in the Time Correction message shall be used to correct the data time stamp and make the time relative to the multi-cast consumers clock.	2-4.4.1
MCast_Byte	Consisting of:	
Parity_Even	FRS62 Parity_Even in the Time Correction message shall be the even parity of bits 0 through 6 of the Message	
Multi_Cast_Active_Idle	FRS358 Multi_Cast_Active_Idle in the Time Correction message shall indicate Idle if transmitted before this consumer has responded with a valid time coordination message <sup>2</sup>	2-4.4.3
Consumer_#	FRS65 The Consumer_# in the Time Correction message shall indicates the number of the consumer that this message is directed to.	2-4.4.1
Reserved	Reserved, set to zero	2-4.4.4
CRC-S3	CRC covering the data within the Base Format time coordination message	
CRC-S5	CRC covering the data within the Extended Format time coordination message	

<sup>2</sup> example code does not transmit Time Correction to a consumer until the initial Time Coordination is received



FRS66 The following rule shall be used for the calculation of the parity bit in the Time Correction message:

Bit 7 of the MCast\_Byte form even parity on the MCast\_Byte.

FRS67 The following rule shall be used for setting of the bits in Mcast\_Byte\_2 in the Time Correction message:

$\text{MCast\_Byte\_2} = ((\text{MCast\_Byte} \text{ XOR } 0xFF) \text{ AND } 0x55) \text{ OR } (\text{MCast\_Byte} \text{ AND } 0xAA)$

FRS371 When a connection is established with the Extended format, the EF Time Correction format shall be used; otherwise the base format shall be used .

### 2-1.7.1.8.3 Time Correction CRC Calculation

FRS68 The Time Correction CRC calculation shall include:

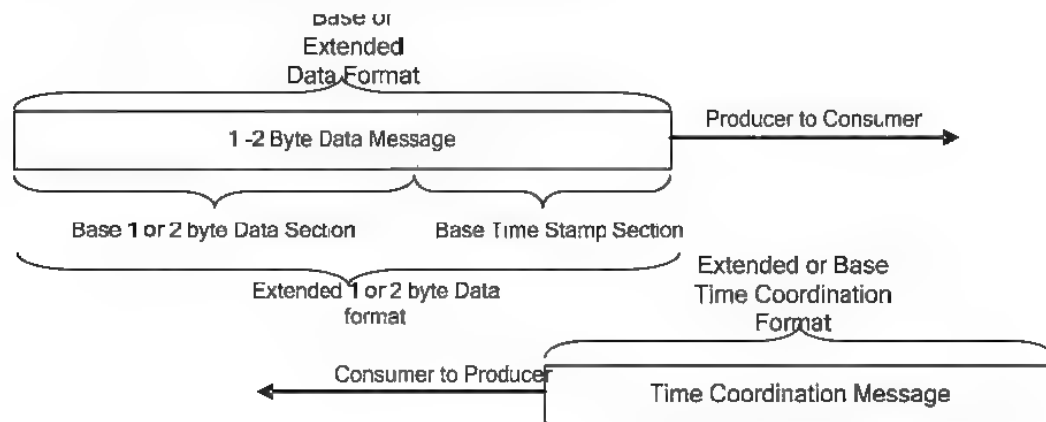
- Producer Identifier (PID) (by CRC-S3 or CRC-S5)
- Mcast\_Byte (by CRC-S3 or CRC-S5)
- Time\_Correction\_Value (by CRC-S3 or CRC-S5) (see light grey colored sections)

FRS69 MCast\_Byte\_2 of the Time Correction message shall not be included in the Safety CRC .

### 2-1.7.1.9 1 or 2 Byte, Single-Cast, Safety Connection Format

Figure 2-1.18 shows the section organization for 1 or 2 byte single-cast safety connection format.

Figure 2-1.18 1 or 2 Byte Single-Cast Message Encoding

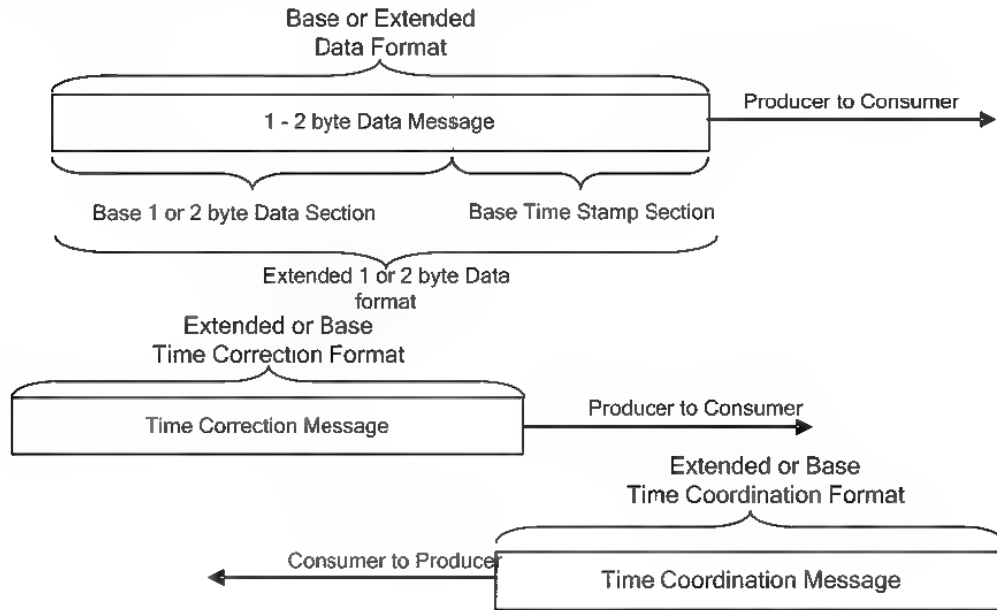


### 2-1.7.1.10 1 or 2 Byte, Multi-Cast, DeviceNet Safety Connection Format

Figure 2-1.19 shows the messages and section organization for the 1 or 2 byte Multi-cast Safety connections when the connections are on DeviceNet.



Figure 2-1.19 1 or 2 Byte Multi-Cast Message Encoding

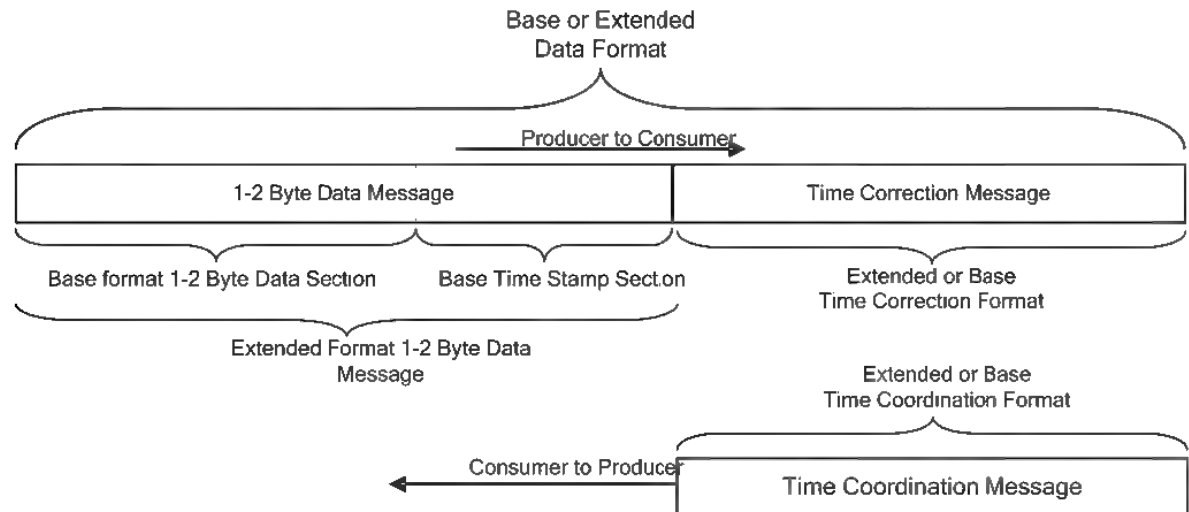




### 2-1.7.1.11 1 or 2 Byte, Multi-Cast, Non-DeviceNet, Safety Connection Format

Figure 2-1.20 - shows the section organization for 1 or 2 byte multi-cast safety connections when the connections are on EtherNet/IP and other Non-DeviceNet networks in general.

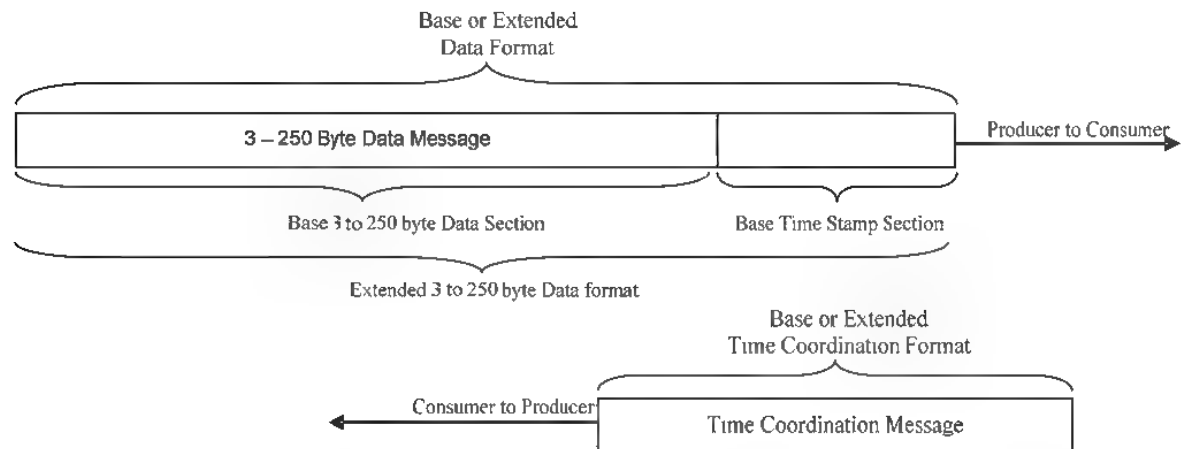
**Figure 2-1.20 1 or 2 Byte, Multi-Cast, Safety Connection Format**



### 2-1.7.1.12 3 to 250 Byte, Single-Cast, Safety Connection Format

Figure 2-1.21 shows the section organization for 3 to 250 byte single-cast safety connection format.

**Figure 2-1.21 3 to 250 Byte Single-Cast Message Encoding**

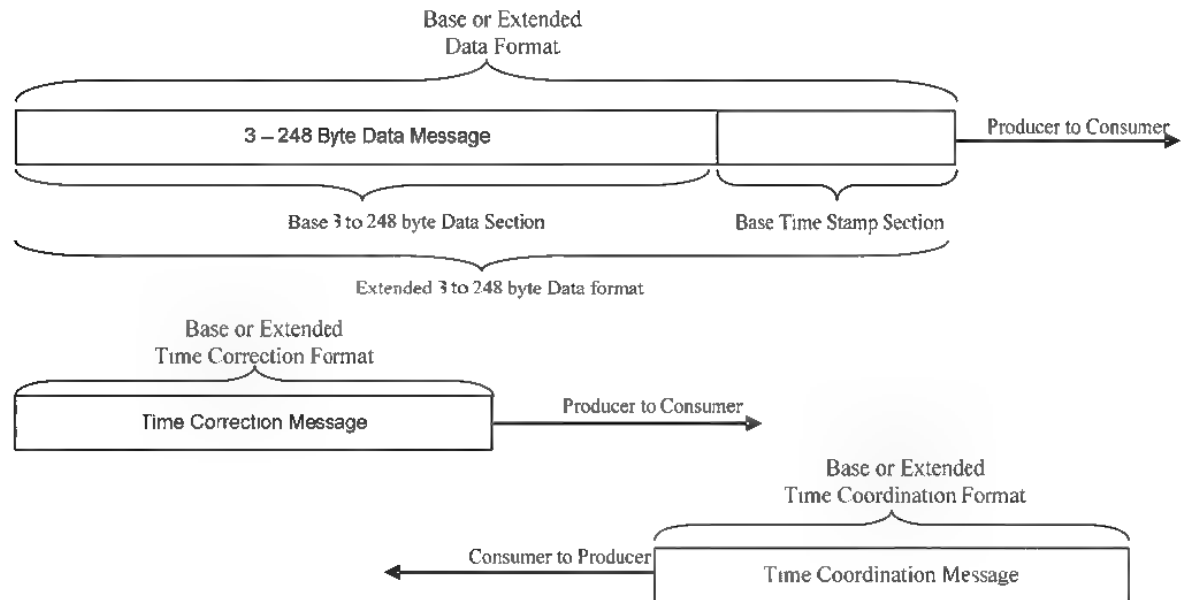




### 2-1.7.1.13 3 to 248 Byte, Multi-Cast, DeviceNet Message Format

Figure 2-1.22 shows the messages and section organization for 3 to 248 byte multi-cast safety connection format when the connections are on DeviceNet.

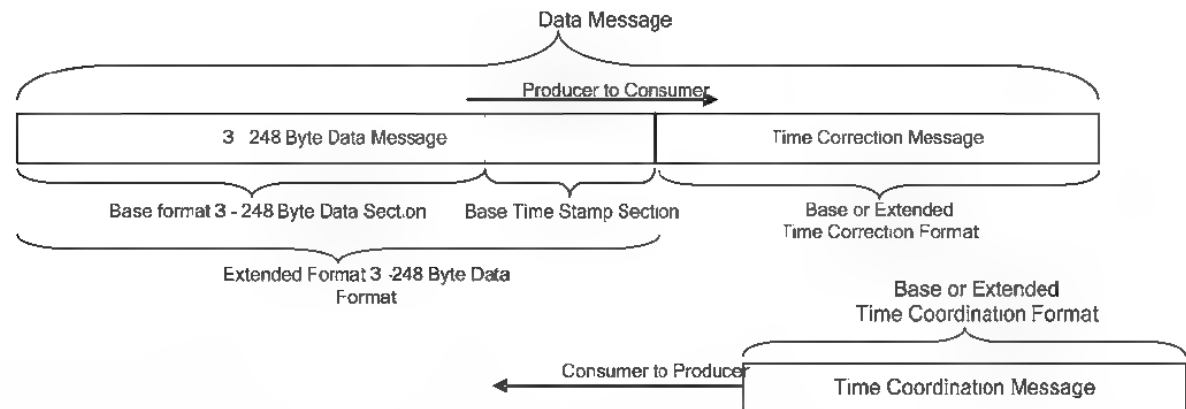
**Figure 2-1.22 3 to 248 Byte Multi-Cast Message Encoding**



### 2-1.7.1.14 3 to 248 Byte, Multi-Cast, Non-DeviceNet, Safety Connection Format

Figure 2-1.23 shows the messages and section organization for 3 to 248 byte multi-cast safety connection format when the connections are on EtherNet/IP or other Non-DeviceNet networks in general.

**Figure 2-1.23 3 to 248 Byte, Multi-Cast, Safety Connection Format**





### 2-1.7.2 Safety CRC

There are five safety CRCs used in the various sections of the safety protocol. An 8-bit CRC (CRC-S1 or CRC-S2) is used for the base format data sections containing up to two bytes of safety data and the time stamp sections. A 16-bit CRC (CRC-S3) is used for base and EF data sections for 3 to 250 bytes of data and for the base format time correction and time coordination messages. A 24-bit CRC (CRC-S5) is used for the EF data format up to two bytes of safety data and for the EF complement data sections for 3 to 250 bytes of data. The 24-bit CRC is also used for the time correction and time coordination messages in Extended Format. A 32-bit CRC (CRC-S4) is used for connection establishment, configuration, and the Safety extension to the EDS file definition.

## 2-1.8 Safety Procedures

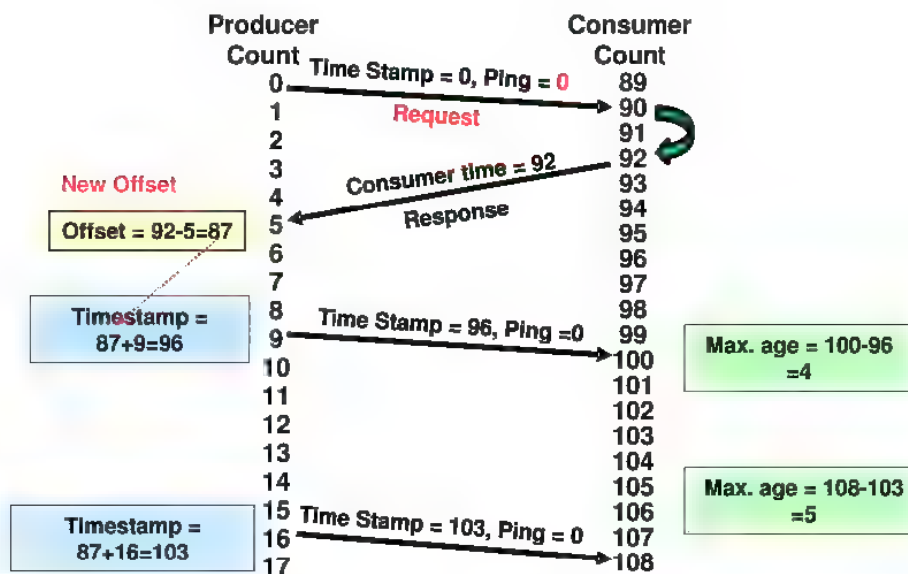
The safety procedures provide a service that ensures data can be transmitted across a network with a defined level of integrity. These procedures are defined as a safety protocol layer using a producer/consumer structure. All safety I/O transmissions across a network or bus will use this safety service.

The usage of safety producers and safety consumers allows for consistency across all safety system topologies.

### 2-1.8.1 Time Stamp Operation

The time stamp allows for the calculation of the timeliness of receipt of data in the consumer. Figure 2-1.24 shows the time stamp sequence of operations for a simple single-cast producer consumer pair.

Figure 2-1.24 Time Stamp Sequence





FRS70 Upon connection establishment, the producer shall request an initial ping response.

FRS71 When the consumer sees a change in the ping count, the consumer shall prepare a Time Coordination response message that contains the consumer's value of its clock count .

FRS72 The producer shall use the clock count in the Time Coordination response to calculate an offset that will be applied to future productions.

FRS73 At each scheduled production, the producer shall use the value of its own clock count and the offset to calculate the time stamp and sends it to the consumer along with the data.

FRS74 The consumer shall use the value received from the producer to calculate the age of the data by subtracting the producers time stamp from its current clock count. If the calculated delta is less than the maximum delta specified by the user, the data is ok and the data is used by the application within the device. If the delta is greater than the user specified value, the data is deemed to be too old to use and the connection is terminated.

FRS75 Once each Ping Interval, the producer shall request another ping response from the consumer. This will cause a recalculation of producer offset and prevent the slow aging of data due to time errors.

A time correction factor is added to account for the maximum skew and drift of clocks in an asynchronous system. This message is used to adjust the offset in the producer.

### **2-1.8.2 Rollover counts in the Extended Format**

A rollover count is employed in conjunction with the transmitted Time Stamp to extend the effective operating range of the Time Stamp to 32 bits. The Rollover count is not transmitted with the time stamp however. The initial values for the rollover count are set at connection establishment and maintained independently by both producer and consumer. The rollover count value is included in the data CRC calculation so both ends must agree on the current value at all times for packets to be received successfully.

### **2-1.8.3 Protocol Sequence Diagrams**

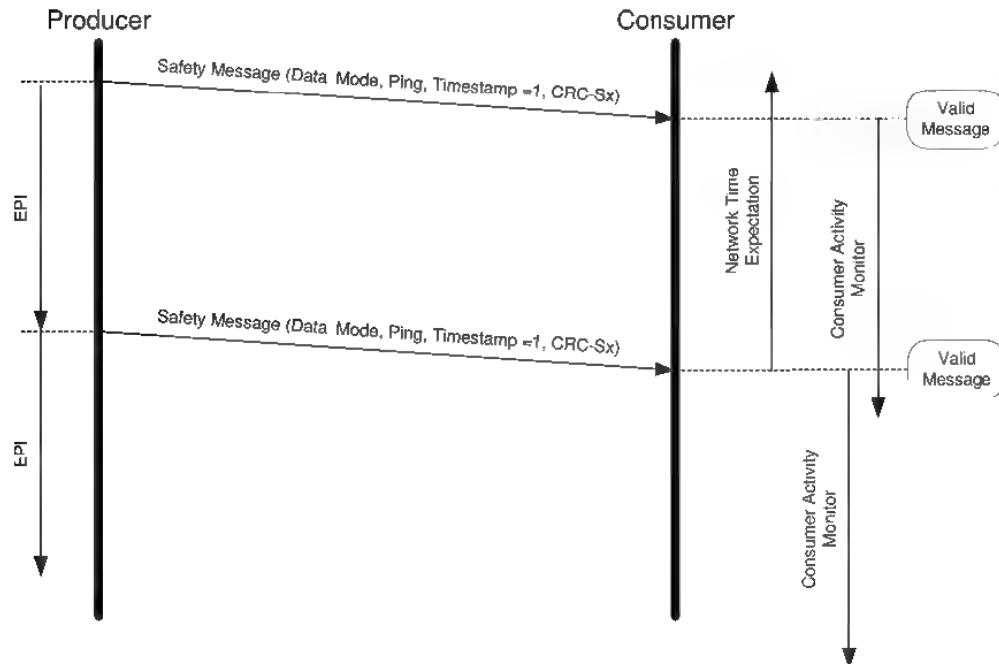
The protocol sequence diagrams show the relationship between producers and consumers with respect to time and events. Time flows from the top of each figure to the bottom. Normal messages that are received at the producer or consumer are shown in normal text boxes. Anomalous messages are shown in bolded text boxes. Messages that were expected but not received are shown as unboxed text.

#### **2-1.8.3.1 Normal Safety Transmission**

Normal communication between a safety producer and consumer requires a sequence of periodic messages. When the consumer receives a valid message, the transaction is complete as shown in Figure 2-1.25.



Figure 2-1.25 Sequence Diagram of a Normal Producer/Consumer Safety Sequence



The safety producer produces data at a fixed rate called the Expected Packet Interval (EPI).

FRS78 Link-triggered consumers shall maintain an activity timer, which is reset after receiving a valid expected message.

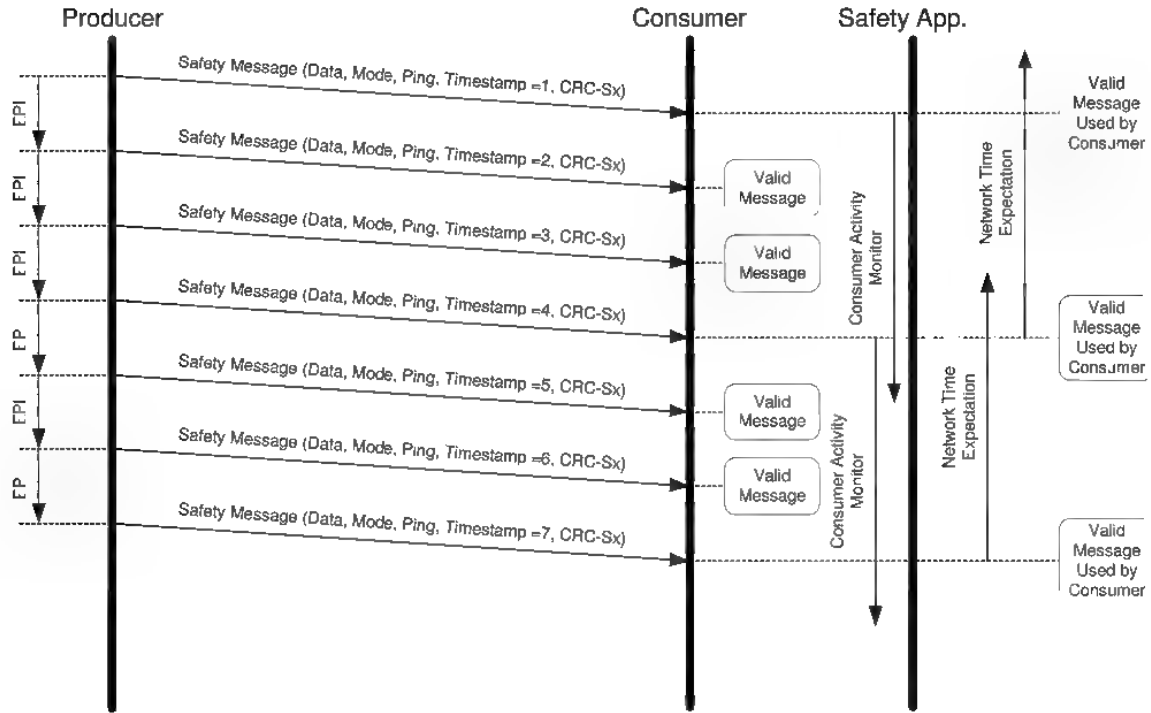
FRS79 If a valid message has not been received before the activity timer expires, the link-triggered consumer shall terminate the connection (See 2-1.3.2) and go to the defined safety state.

FRS80 All consumers shall check the age of the data received by checking the time stamps received.

FRS81 If the age of the data is greater than the network time expectation the consumer shall terminate the connection (See 2-1.3.2) and go to the defined safety state.



**Figure 2-1.26 Sequence Diagram of a Normal Producer/Consumer Safety Sequence (production repeated)**



FRS82 Safety communications shall be configurable with production repeated. In this case, the consumer does not use every packet that is sent to it.

FRS83 When safety communication is sent with repeated production, application-triggered consumers shall use the last packet received to complete the transaction. , as show in Figure 2-1.26

FRS84 Consumers using base format connections shall see the Timestamp increase for each successive period or the connection shall be terminated . (See 2-1.3.2)

FRS372 Consumers using Extended Format connections shall see the Timestamp increase for each successive period. If the consumer receives a packet with an out-of-sequence Timestamp, it shall drop the packet unless the Consumer Fault Counter has reached the Max Fault Number; in which case the connection shall be terminated. (See 2-2.6.3.1)

Messaging with production repeated offers the additional advantage of using the most current data.

### 2-1.8.3.2 Lost, Corrupted and Delayed Message Transmission

Due to external disturbances it may be possible for the message to be corrupted as it is transmitted from the producer to the consumer, as shown in Figure 2-1.27.

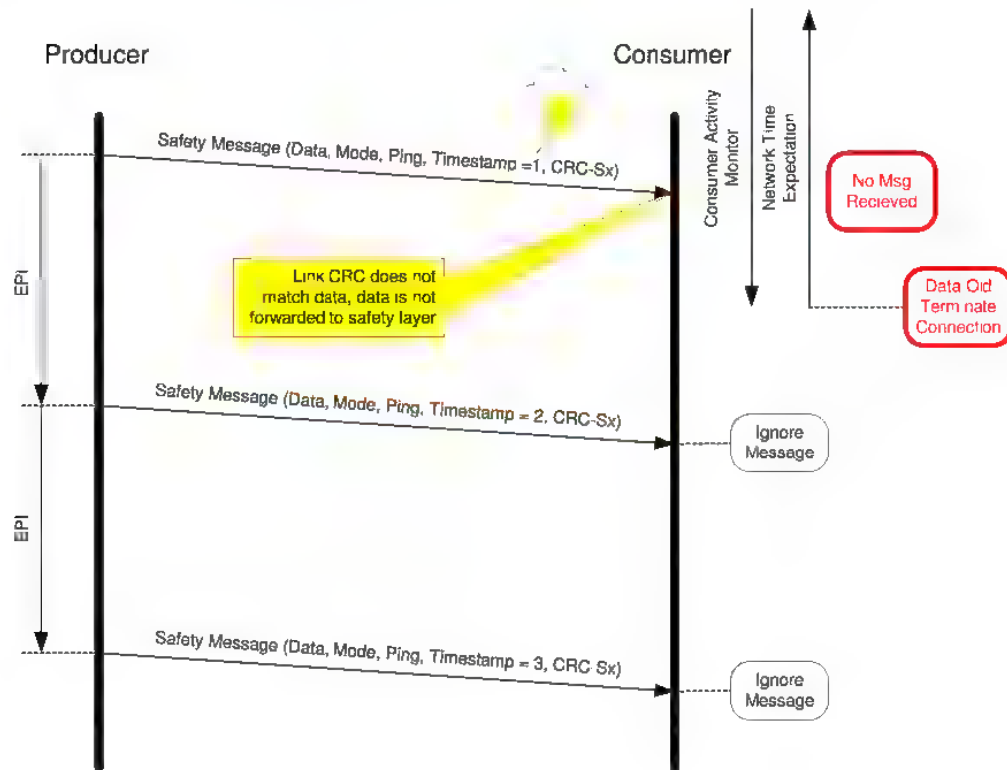
FRS85 If a corrupted message is detected at the standard layer, it shall be discarded and not presented to the application.



This will result in an expiration of the consumer activity monitor or the data age will exceed the network time expectation and the consumer terminates the connection (See 2-1.3.2) and goes to the defined safety state.

FRS86 If the transmitted CRC-Sx and data are checked and any found to be in error or the data cross-check is found to be in error, the consumer shall either drop the packet (Extended Format AND Consumer\_Fault\_Count less than Max\_Fault\_Number), or terminate the connection and go to the defined safety state (Base format or Extended Format AND Consumer\_Fault\_Count greater or equal to Max\_Fault\_Number). (See 2-1.3.2)

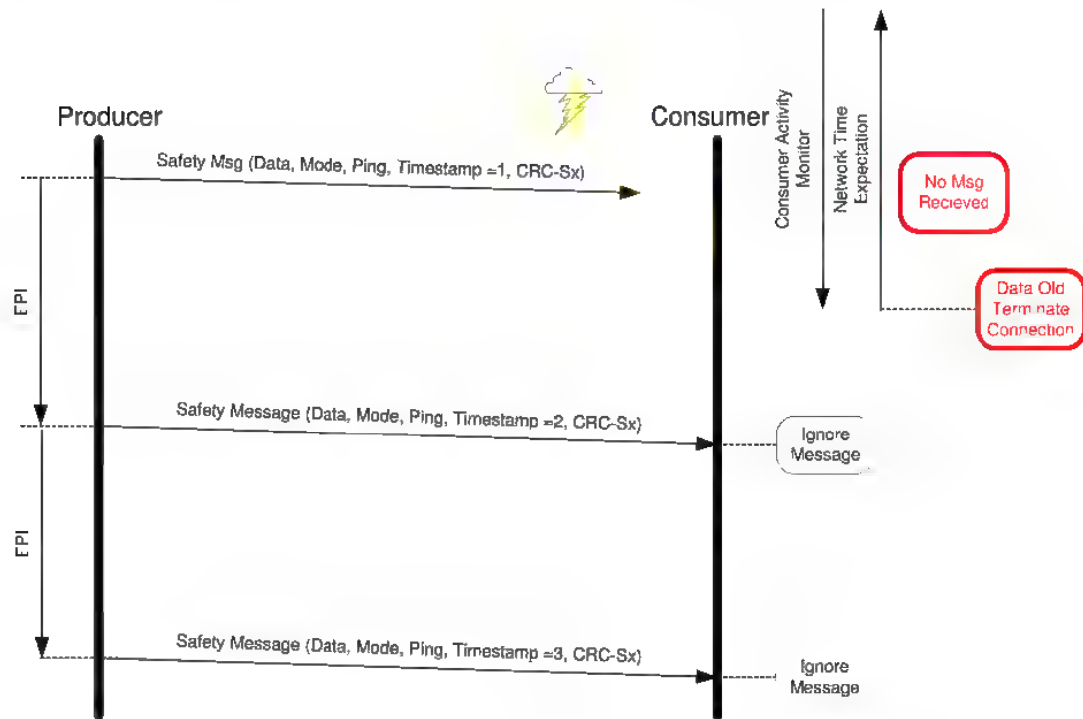
Figure 2-1.27 Sequence Diagram of a Corrupted Producer To Consumer Message



Due to external disturbances it may be possible for the message to be lost as it is transmitted from the producer to the consumer, as shown in Figure 2-1.28. This will result in an expiration of the consumer activity monitor or the data age will exceed the network time expectation and the consumer terminates the connection (See 2-1.3.2) and goes to the defined safety state.



Figure 2-1.28 Sequence Diagram of a Lost Producer to Consumer Message



Messages may be delayed in transmission. FRS87 If a message is delayed such that it is received

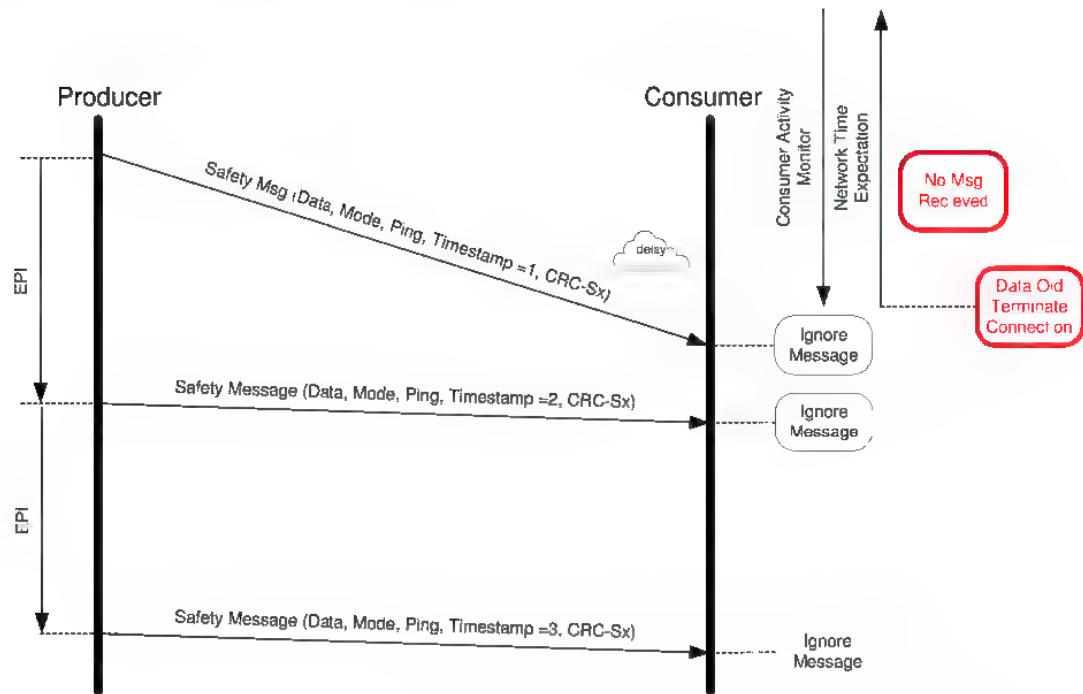
1. beyond the consumer activity monitor (link-triggered application) or
2. the data age exceeds the network time expectation,

then all further messages are ignored, and the connection shall be terminated.

In both cases the consumer terminates the connection (See 2-1.3.2) and goes to the defined safety state. This behavior is shown in Figure 2-1.29.



Figure 2-1.29 Sequence Diagram of a Delayed Message

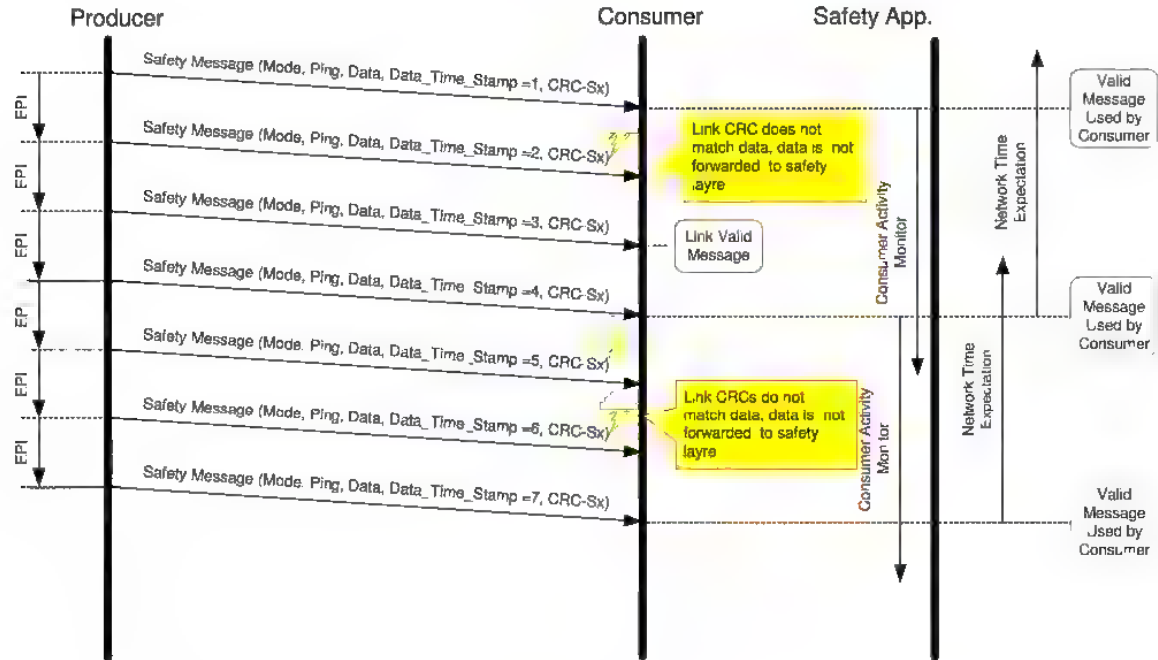


### 2-1.8.3.3 Lost, Corrupted or Delayed Message Transmission with Production Repeated

Due to external disturbances it may be possible for the message to be corrupted as it is transmitted from the producer to the consumer, as shown in Figure 2-1.30. In order to increase the availability of the overall system, production may be repeated.



**Figure 2-1.30 Sequence Diagram of a Corrupted Producer to Consumer Message With Production Repeated**



If there are errors that are detected in the standard layers, the messages can be discarded, as long as a valid and expected message is received within the consumer activity monitor and the age of the data does not exceed the network time expectation.

FRS88 Valid repeated messages shall be overwritten. The most recent valid message is used by the application.

Due to external disturbances, it may be possible for a series of messages to be delayed resulting in a lost connection. This sequence is shown in Figure 2-1.31. If no valid messages are received before the consumer periodic timer expires, the connection will be terminated (See 2-1.3.2).





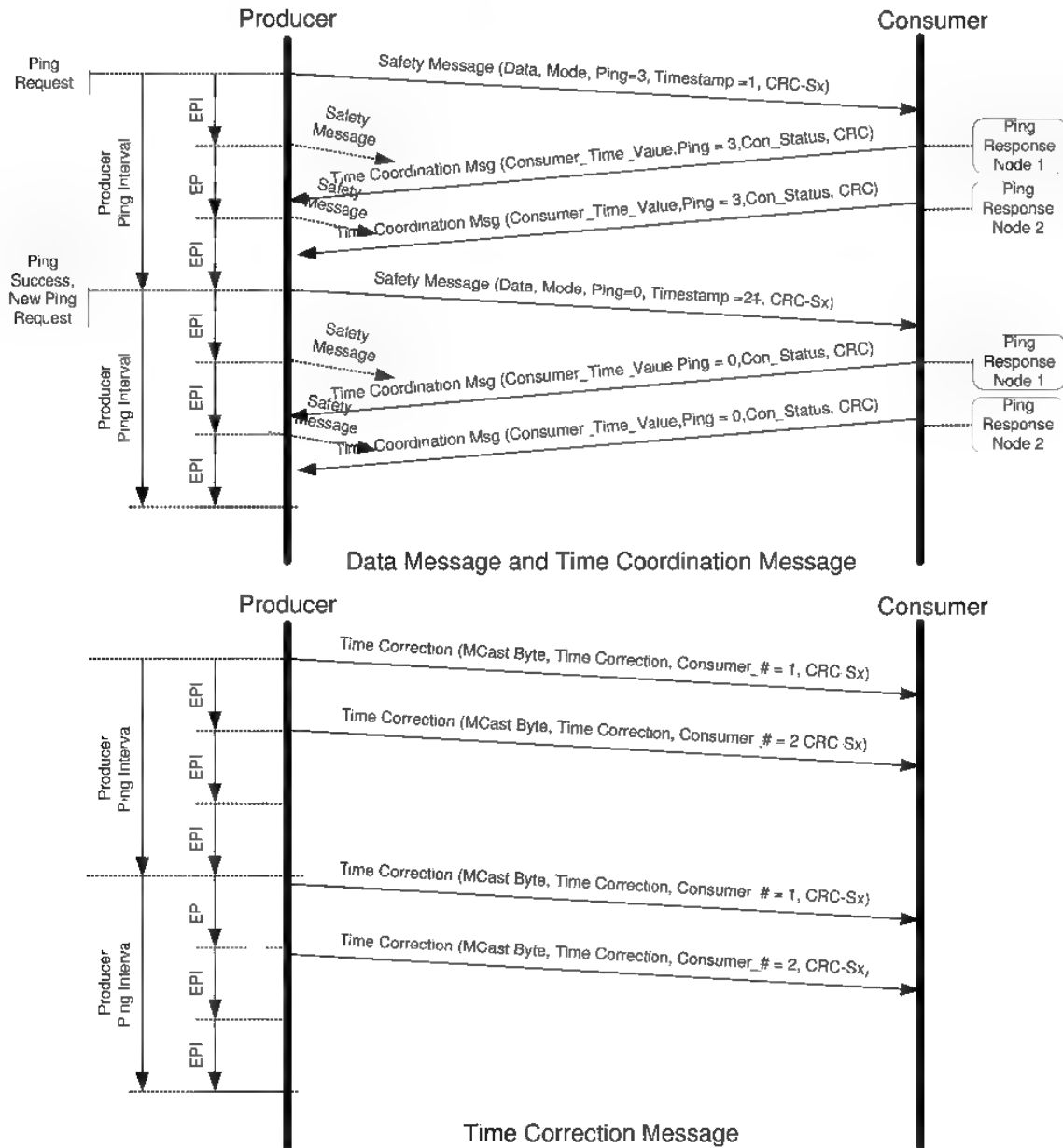






The producer is fully described in section 2-2.4 and the consumer is fully described in section 2-2.6. This sequence of messages for multi-cast ping is shown in Figure 2-1.34. For clarity the time correction message is shown on a separate diagram.

Figure 2-1.34 Sequence Diagram of a Successful Multi-Cast Ping, DeviceNet



During each multi-cast producer ping interval, a ping request will be sent from the multi-cast producers to multi-cast consumers requesting a response.

FRS94 All multi-cast consumers shall respond to a change in the ping request value, consumers other than consumer 1 shall wait Consumer\_Number -1 times the EPI to reply.



All of the multi-cast consumers must respond within the ping interval of the request or the next one.

FRS95 Time correction messages shall be sent to the multi-cast consumers at a rate to ensure that each consumer gets a time correction message within a single producer ping interval

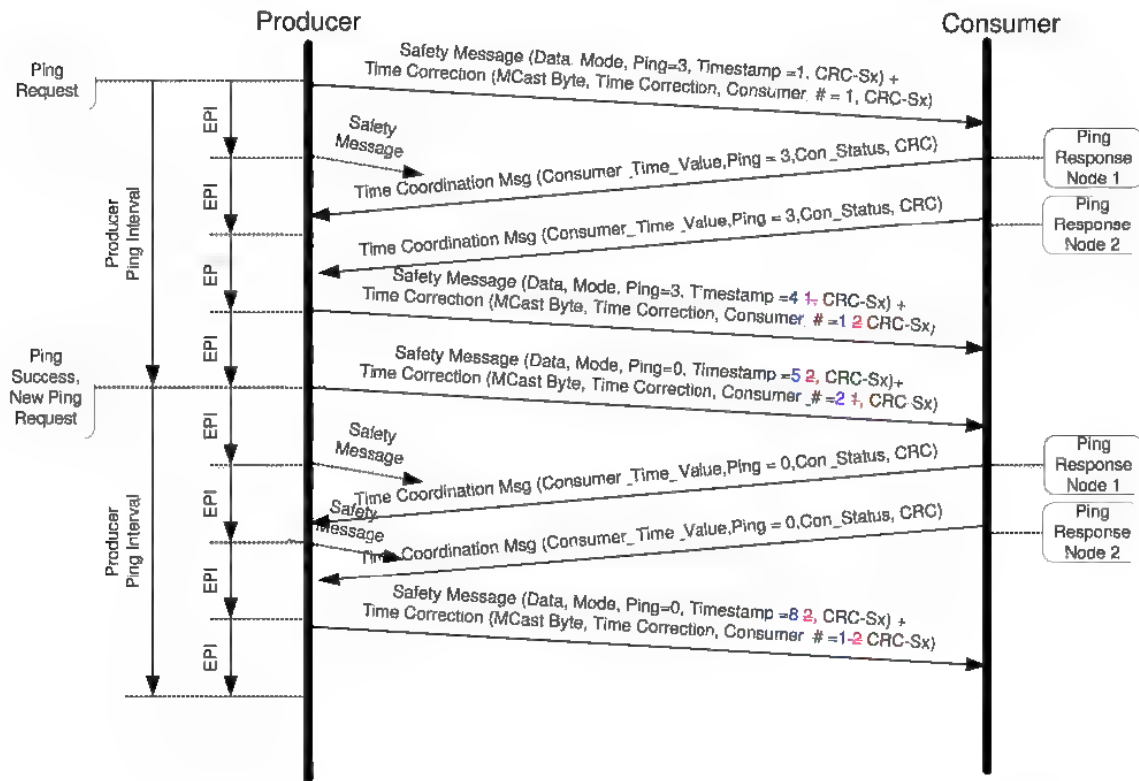
### 2-1.8.3.6 Multi-Cast Ping On Non - DeviceNet Networks

In order to maintain delay timing between multi-cast producers and its consumers, a ping request and response is used.

FRS96 For Non-DeviceNet networks the time correction message shall be concatenated with the safety data message and sent as a single message.

The producer is fully described in section 2-2.4 and the consumer is fully described in section 2-2.6. This sequence of messages for multi-cast ping is shown in Figure 2-1.35.

Figure 2-1.35 Sequence Diagram of a Successful Multi-Cast Ping, Non-DeviceNet



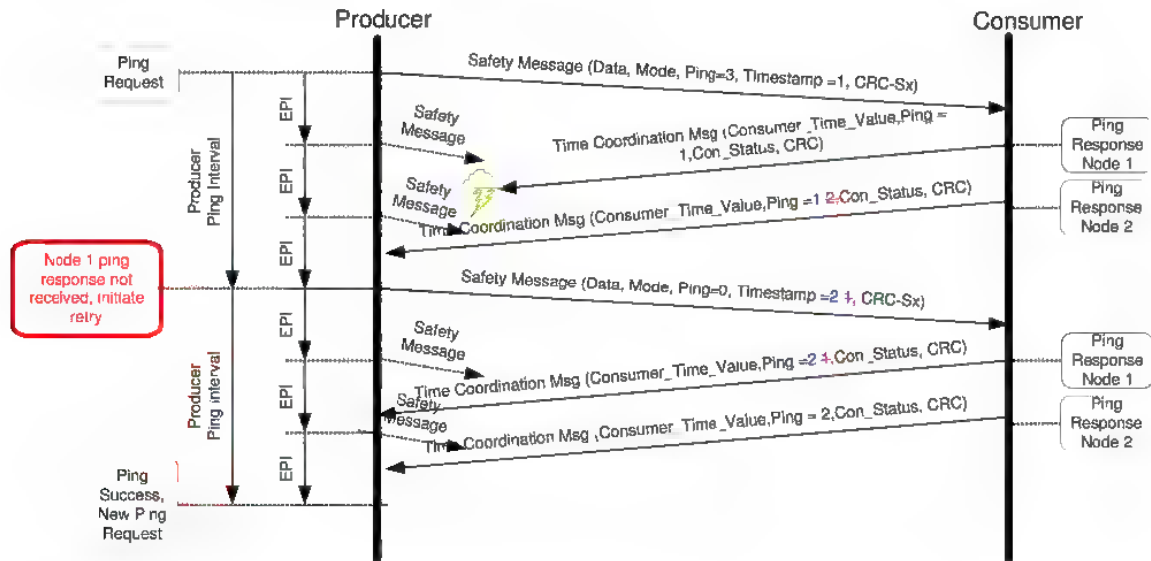
During each producer ping interval, a ping request is sent from the multi-cast producers to multi-cast consumers requesting a response.



### 2-1.8.3.7 Multi-Cast Ping – Retry with Success

This sequence of messages for a multi-cast ping with retries is shown in Figure 2-1.36. The time correction connection is not shown for clarity.

**Figure 2-1.36 Sequence Diagram of a Multi-Cast Ping Retry**



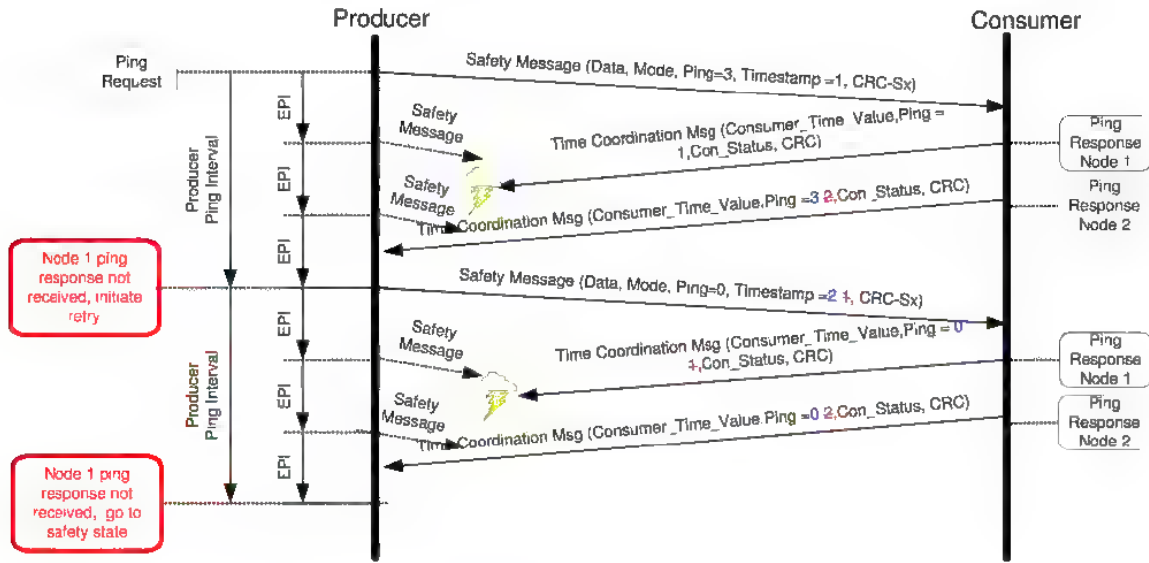
FRS98 If a multi-cast time coordination message is lost or delayed past the (Timeout\_Multiplier.PI +2) ping intervals, an error shall be generated and the connection closed .

### 2-1.8.3.8 Multi-Cast Ping – Retry with Timeout

This sequence of messages for a multi-cast ping with retries and timeout is shown in Figure 2-1.37. The time correction connection is not shown for clarity.



Figure 2-1.37 Sequence Diagram of a Multi-Cast Ping Timeout



If the ping response continues to be delayed, the producer will indicate a fault in the producer status for the specific consumer and the consumer will go to the defined safety state. If the ping was delayed due to a fault in the consumer, the consumer will have already gone to a safe state.

## 2-1.9 Safety Configuration

This section gives an overview of the configuration procedure and data. Complete details of the safety configuration process are given in Chapter 7 of this volume.

There are three methods identified for configuring safety devices:

Configuration as part of the forward open 2-1.9.1.2

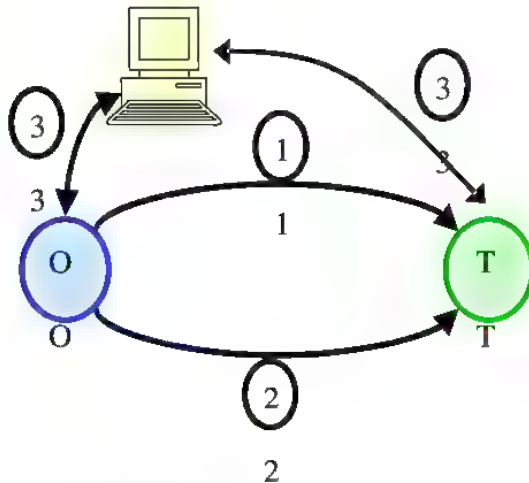
Configuration by separate transfer from an originator 2-1.9.1.2

Configuration from a workstation 2-1.9.1.1

The relation ship of these configuration processes is shown in Figure 2-1.38



Figure 2-1.38 Methods of Configuring Devices



### 2-1.9.1 Safety Configuration Data

Safety devices will require configuration data be stored and maintained for both the device configuration and the connection configuration.

FRS100 It shall be permissible to combine the device and the connection configuration.

The overall safety configuration is specific to the network and application being used and may require more data (i.e. configuration by an application configuration tool and a network configuration tool). The connection configuration information is defined in Chapter 6.

#### 2-1.9.1.1 Configuration from Workstation

The process for configuration of a safety device from a workstation is fully described in the Chapter 7. When either an originating device or a target device is configured from a workstation, the modes of the devices and the process of configuring a device shall be managed such that the device will meet the desired SIL level requirements upon the completion of the configuration process.

#### 2-1.9.1.2 Configuration as Part of Connection Establishment

The overview of the operations required to initially configure device connections is described in this section. A complete definition of connection establishment is found in Section 2-3.

#### 2-1.9.1.3 Types of safety opens

There are two types of safety opens defined, Type 1 SafetyOpen and Type 2 SafetyOpen.



A Type 1 SafetyOpen has all of the data necessary for the configuration of a safety device. A Type 2 SafetyOpen does not contain all of the data necessary for the configuration of a safety device. Type 2 SafetyOpens may perform a SCID check to ensure that the originator and target are in agreement on the configuration data of the target depending on the value of the SCID (refer to FRS163). If a Type 2 SafetyOpen is used with the option to not perform the SCID check, instructions shall be placed in the user manual for the user to check that configurations used are as expected. (refer to FRS103)

A Type 1 SafetyOpen shall not be limited to EDS configurations. Any settable attribute data, regardless of the attribute's relation to safety, may be included within a Type 1 SafetyOpen.

#### **2-1.9.1.3.1 Configuration Using a Type 1 SafetyOpen**

At the time a connection is established, the originator sends a forward open with a safety network segment on EtherNet/IP and SERCOS III or SafetyOpen on DeviceNet that contains the originator UNID, the target UNID, the Safety Configuration Identifier (the configuration data is omitted for Type 2 SafetyOpens). The target device verifies the target UNID matches its UNID and verifies ownership, validity of data and Safety Configuration Identifier match prior to applying the configuration data.

FRS104 Once the producer has correctly verified a successful response to a forward open message, the producer shall start producing safety data at the periodic rate but with the run/idle bit set to idle (single-cast) or the Consumer\_Active\_Idle set to Idle (multi-cast).

The producer shall send a ping request on the first message.

FRS106 The producer shall maintain the run/idle or Consumer\_Active\_Idle bit idle until the initial Time Coordination sequence is successfully completed.

FRS107 The consumer shall remain in a safety state until the initial ping sequence is successfully completed.

The consumer will know the sequence is successful when there are no faults indicated and the Run\_Idle bit or Consumer\_Active\_Idle bit are set to the run state. A detailed explanation of these interactions is contained in Section 2-6, Safety Connection Establishment.

#### **2-1.9.1.3.2 Configuration Using a Type 2 SafetyOpen**

FRS108 If a device either responds to a Type 2 SafetyOpen with a SCID mis-match or requests a download because it is being replaced, the following sequence shall be followed.

- The originator sends a Type 2 SafetyOpen to the target device
- The target responds with a mode state error
- The originator calls the Safety Application to perform a download
- The Safety Application performs and verifies the download.
- The download may be a series of commands to the device or
- The download may be a Type 1 SafetyOpen.
- The originator repeats the Type 2 SafetyOpen to establish the connection verify the configuration.



### **2-1.9.2 Safety Network Configuration Tool**

The safety network configuration tool (SNCT) shall support the functionality outlined in this section.

### **2-1.9.3 Safety Configuration Data CRC (SCCRC)**

The SCCRC is defined in Section 2-3.1.1.

### **2-1.9.4 Safety Configuration Identifier (SCID)**

All safety configurations shall be accompanied by a Safety Configuration Identifier to identify the configuration. The format and contents of the Safety Configuration Identifier is defined in Section 2-3.1.1

FRS111 The SNCT which generates a safety device configuration shall generate a unique Safety Configuration Identifier .

SRS47 The SCID shall change when a device's configuration data changes.

### **2-1.9.5 Device Replacement**

FRS112 The user safety manual shall contain instructions instructing the user "The replacement of safety devices requires that the replacement device be configured properly and operation of the replacement device shall be user verified.

#### **2-1.9.5.1 Devices Configured from SafetyOpen/Forward Open**

If a device configured via a safety open is replaced, the originator will attempt to re-establish the connections and configure the new device. The actions necessary for these operations are detailed in section 2-3.

When the safety open is received by the new device, the Safety Configuration Identifier will not match the Safety Configuration Identifier contained in the SafetyOpen. Once the SafetyOpen TUNID is validated as correct and the safety data in the SafetyOpen is validated than the configuration is applied by the device.

### **2-1.10 Diagnostics for Communication Errors**

Safety Devices on CIP Networks should support any diagnostic features specified in the CIP Common Specification (Volume 1), and the appropriate network specific specification, either the DeviceNet Specification (Volume 3) or the EtherNet/IP Specification (Volume 2). Communication error diagnostics for Safety Devices on SERCOS III networks is outside the scope of this specification.

#### **2-1.10.1 Rules for Device Configurations**

FRS116 The following Rules shall apply to device configuration:



- Devices being configured shall not have any active safety connections and shall not be in the executing mode.
- Once configuration of a device starts, the device shall remain in the non-operational mode until the configuration process is successfully completed. This mode shall be maintained through power cycles.
- If a device is in an operational mode (e.g., has active connection or is in executing mode), it shall reject any configuration messages.

#### **2-1.10.2 Connection Restart**

A restart may be needed if a connection is terminated for a safety event or terminated from the application. It may also be desirable to restart a single connection rather than the entire connection set for a device. The restart sequence is the same message sequence from the originator to the target as defined in Section 2-3. Due to the nature of some of the errors, it may require a power cycle or module replacement before the error is cleared.

#### **2-1.11 Safety Device Requirements (Device Behavior)**

The safety device requirements define the behavior that all devices need to implement shall support proper operation of the safety protocol.

##### **2-1.11.1 Safety Input Requirements**

FRS117 If a safety state exists for the safety-input device, the device shall transmit safety data and the Run\_Idle bit shall be set to the idle state upon detection of a fault by the input circuit diagnostics.

FRS118 Changes in the state of the physical process inputs of a safety device shall be transmitted at the next available periodic update time.

##### **2-1.11.2 Output Requirements**

All safety outputs will go to a safety (off) state on detection of a failure.

Safety output devices may provide the output point status. If output status is needed it should be sent via a single-cast or multi-cast input safety connection.

##### **2-1.11.3 Safety Device Communications Failure**

FRS120 A failure in a background communications diagnostic shall cause all producer/consumer safety connections for that device to be terminated. (See 2-1.3.2)

##### **2-1.11.4 Safety Device Major Faults**

FRS121 If there is a major fault in a safety device that causes the device to go into a safety state, the device shall be reset or restarted.

FRS122 In order to clear a major fault in a system the device shall execute and pass all internal self-tests prior to accepting or initiating any safety connections.



#### **2-1.11.5 Safety-Related Program Detected Safety Event**

It is expected that if a safety-related program detects a safety event, the safety-related program will take a safety action, applying required safety data to the outputs.

#### **2-1.11.6 Safety Device Loss of Power**

A loss of device power in a safety device will cause the producer/consumer safety connections for that device to terminate. The powering up device will then follow standard safety start up procedures to rejoin the system.

### **2-2 Safety Protocol**

The safety layer is a layer of service residing between the standard CIP transport layer and the safety application layer as shown in Figure 2-1.2. This safety layer need not be directly addressable in the CIP framework; rather it is a behavior associated with a connection between applications exchanging data in a SIL 3 application. This behavior is implemented, in some products, as the Safety Validator object as described in section 2-2.2.

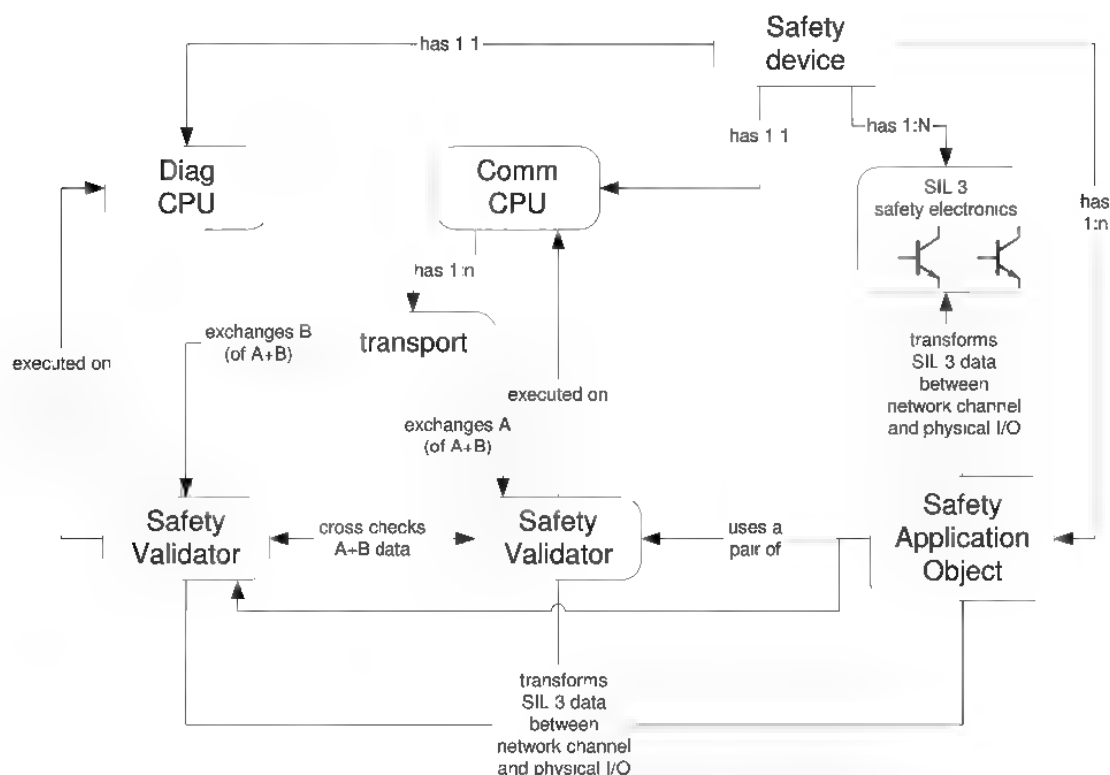
Note: There may be multiple methods of accomplishing this desired behavior. The objects in this chapter are to be considered a reference model. Individual products and applications may choose to implement this behavior, as they deem necessary. However, this behavior needs to be expressed here to form a basis for product development.

#### **2-2.1 High Level View of a Safety Device**

Figure 2-2.1 shows a reference model of a safety device containing one or more safety application objects. Each safety application objects functions with a pair of Safety Validator objects to exchange SIL 3 data between the CIP transport layer and the safety I/O electronic circuitry. These electronics are designed to provide a level of error detection and fault isolation suitable for SIL 3 applications. Each of the two Safety Validator objects is associated with a network connection on networks such as DeviceNet or Ethernet/IP. This model assumes a 1oo2 Safety Topology in which two independent CPUs participate in the safety protocol.



Figure 2-2.1 Safety Device Reference Model Entity Relation Diagram



## 2-2.2 Safety Validator Object

The Chapter 5-5 of this volume defines a Safety Validator object from which both the SafetyValidatorServer and SafetyValidatorClient functions are implemented.

The next section describes the general principles and common attributes of these two safety layer functions.

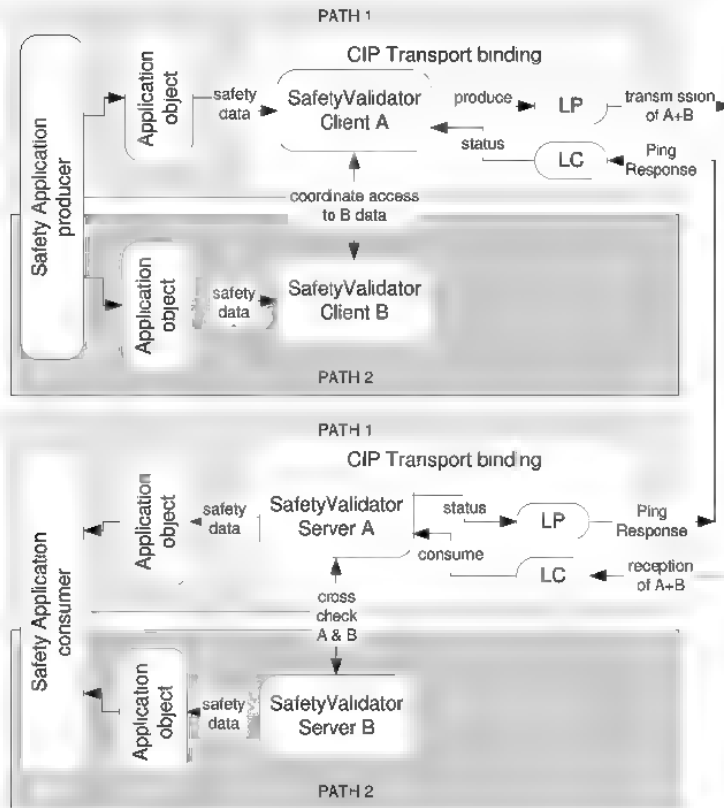
## 2-2.3 Relationship between SafetyValidatorServer and SafetyValidatorClient

Figure 2-2.2 shows a high level view of two devices interchanging data via a SafetyValidatorClient and a SafetyValidatorServer. A safety producing application uses a SafetyValidatorClient to send safety data to a safety consuming application that uses a SafetyValidatorServer.

Note that only one of the redundant paths interfaces to the CIP transport layer. The A+B (actual and complement) data is encapsulated in a single transport frame and only one processor is involved in the reception and transmission of the data frames



**Figure 2-2.2 Two Devices Interchanging Data via a SafetyValidatorClient and a SafetyValidatorServer**



## 2-2.4 Extended Format Time Stamp Rollover Handling

The Extended format requires coordinated synchronization between producers and consumers on connection establishment along with maintenance operations to detect and handle Time Stamp rollover events. The following sections graphically show these operations for each unique case. The example code shows these operations in their proper place in the code as well.

### 2-2.4.1 Single-Cast, Originator Consumer, Target Producer

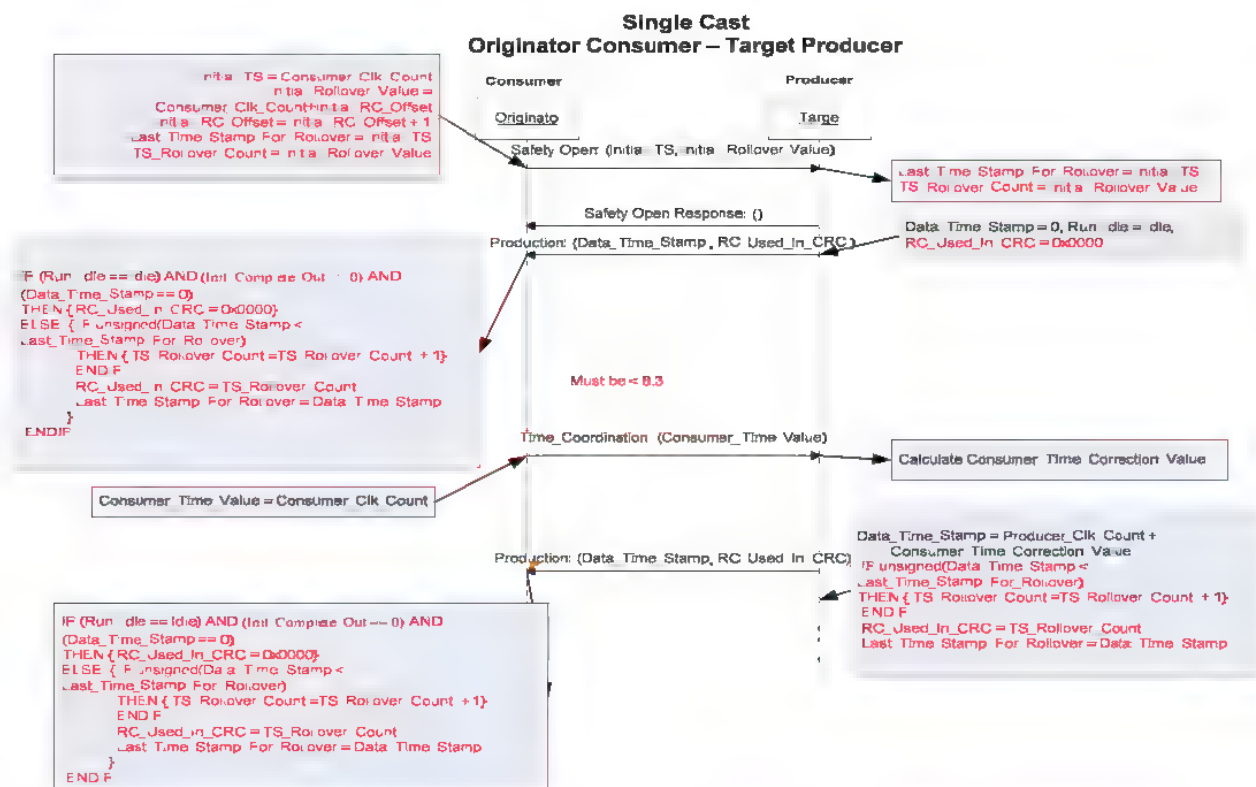
When single-cast connections are originated by the consumer (i.e. single cast inputs), the originator has to provide initialization values in the SafetyOpen. The target uses these values to initialize production parameters.

FRS376 The rollover count used in the Extended Format Single-Cast CRC shall be zero until the initial Time Coordination exchange has been completed.

The complete process is shown below in Figure 2-2.3



Figure 2-2.3 Single-Cast, Originating Consumer Target Producer



The time between the generation of the Safety Open and the Consumers processing of the first packet produced after the producers reception of the Time\_Coordination message must be < 8.3 seconds.

#### 2-2.4.2 Single-Cast, Originator Producer, Target Consumer

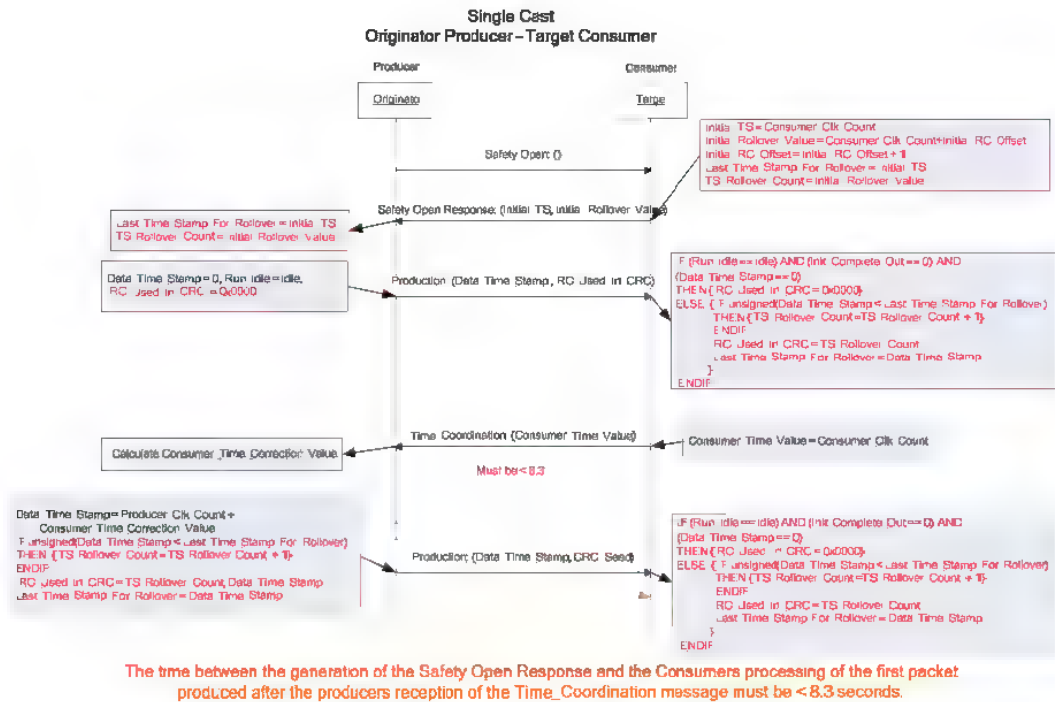
FRS379 When Extended Format single-cast connections are originated by the producer (i.e. outputs), the target consumer shall provide initialization values in the SafetyOpen Response.

The originator producer uses these values to initialize production parameters.

The complete process is shown below in Figure 2-2.4.



Figure 2-2.4 - Single-Cast, Originator Producer, Target Consumer



### 2-2.4.3 Multi-cast, Originator Consumer, Target Producer

FRS377 When Extended Format multi-cast connections are originated by the consumer (i.e. multi-cast inputs); the target producer shall provide initialization values in the SafetyOpen Response.

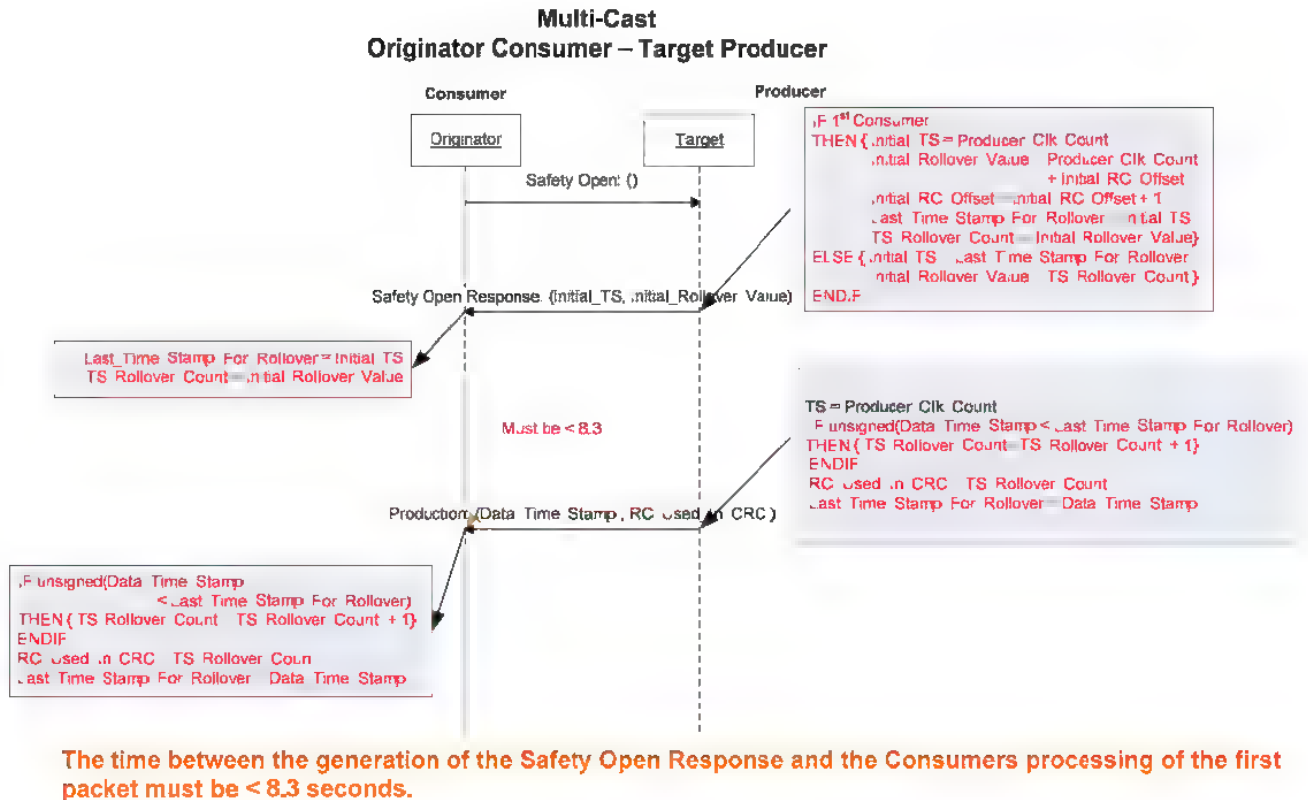
The target consumer uses these values to initialize production parameters. The values the target producer sends in the SafetyOpen Response differs depending on whether this is the 1<sup>st</sup> consumer to request a connection or not.

FRS378 The active seeding of the CRC with the rollover count in Extended Format Multi-cast messages shall begin immediately on first production.

The complete process is shown below in Figure 2-2.5



Figure 2-2.5 - Multi-Cast, Originator Consumer, Target Producer



## 2-2.5 SafetyValidatorClient Function definition

The SafetyValidatorClient is the embodiment of the safety layer that augments a standard CIP connection to transmit SIL 3 data. It coordinates the production of application data with a peer instance of the SafetyValidatorClient.

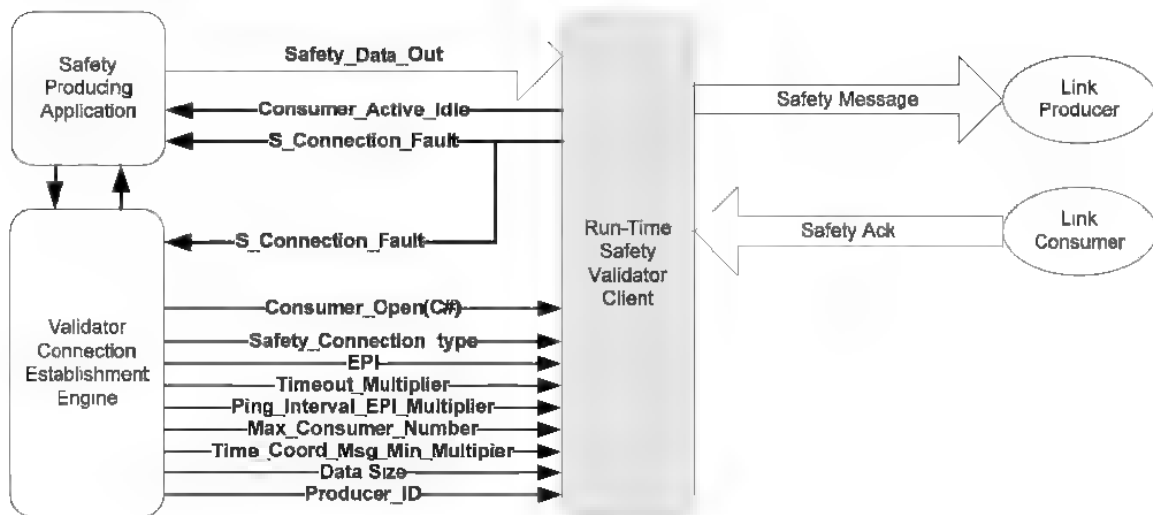
### 2-2.5.1 Safety Production

This section describes the production of data and the interfaces between the application, the SafetyValidatorClient and the underlying link layer. Note that the focus is on the SafetyValidatorClient run-time behavior and its interfaces. The application itself must be of high integrity and this can be accomplished in many ways and is ultimately a vendor design decision; however later in this document, a reference model is presented of an architecture has been qualified for SIL 3 requirements in specific instances. This reference model is for informative purposes only.

FRS123 Safety\_Data\_Out is produced at a periodic rate, referred to as the Expected Packet Interval (EPI). The SafetyValidatorClient shall sample, capture, and time stamp the data to be sent every EPI time period.



Figure 2-2.6 Safety Production Data Flow



### 2-2.5.1.1 Producing Application Interface

FRS124 The producing application provides Safety\_Data\_Out to the SafetyValidatorClient. The SafetyValidatorClient shall build the Mode\_Byte, Actual\_Data, and Complement\_Data, and Time stamp section. (see section 2-1.7.1 for formats)

FRS125 The SafetyValidatorClient shall provide safety connection status back to the producing application for each consumer.

#### 2-2.5.1.1.1 Safety Data Production Logic

This section describes the logic that shall be followed by all safety data producers. The actual implementation of the logic may vary, but equivalent results shall be obtained. The logic in this document assumes an asynchronous producing application that makes the safety data available to the SafetyValidatorClient.

Definitions for the variables used in the safety data production logic can be found in section 2-4.5.

##### 2-2.5.1.1.1.1 Example Safety Data Production Cold Start Logic

The production logic to initiate a cold start of a connection is:

```

////////////////////////////////////
// Cold start after connection establishment processing
////////////////////////////////////
// This Logic should be executed at the transition of the
// producing connection from closed to open.
// For Single-Cast connections this logic may
// be performed any time the connection is Opened or Re-Opened.
////////////////////////////////////
    
```



```

Ping_Interval_EPI_Count = 0,
RR_Con_Num_Index_Pntr = Max_Consumer_Number,

// Initialize the ping count in the safety message to 0
Mode_Byte.Ping_Count = 0,

// Time_Drift Per Ping Interval, the minimum value is 1
Time_Drift_Per_Ping_Interval =
    Roundup(EPI * Ping_Interval_EPI_Multiplier / 320000),

FOR (Consumer_Num = 1 to Max_Consumer_Number),
{
    // Producer Dynamic Variables
    Consumer_Active_Idle[Consumer_Num-1] = Idle,
    S_Connection_Fault[Consumer_Num-1] = OK,
    Producer_Rcvd_Time_Value[Consumer_Num-1] = 0x0000,
    Consumer_Time_Correction_Value[Consumer_Num-1] = 0x0000,
    Ping_Int_Since_Last_Time_Coord_Msg_Count[Consumer_Num-1] = 0x0000,
    Consumer_Time_Value[Consumer_Num-1] = 0x0000,
    Producer_Fault_Counter[Consumer_Num-1] = 0, //For ExtendedFormat only

    // Producer Derived Variables

    // Time_Drift_Constant, the minimum value is 1. (Time_Drift_Constant is not
    // saved but is used in the calculation of the
    // Connection_Correction_Constant below.)
    Time_Drift_Constant =
        Roundup((Timeout_Multiplier.PI [Consumer_Num-1] +1) * EPI *
            Ping_Interval_EPI_Multiplier / 320000),

    // Connection_Correction_Constant
    Connection_Correction_Constant[Consumer_Num-1] =
    Time_Drift_Constant + 1 - Time_Coord_Msg_Min_Multiplier [Consumer_Num-1],

    // Time_Coord_Response_EPI_Limit
    Time_Coord_Response_EPI_Limit[Consumer_Num-1] = Roundup((5000000 +
        (Time_Coord_Msg_Min_Multiplier[Consumer_Num-1]*128) +
        (EPI * (Consumer_Num - 1))) / EPI),

    // Time_Coord_Response_EPI_Limit has a maximum value of 1000
    IF (Time_Coord_Response_EPI_Limit[Consumer_Num-1] > 1000),
    THEN
    {
        Time_Coord_Response_EPI_Limit[Consumer_Num-1] = 1000,
    }
    ENDIF
}
ENDFOR
IF (ExtendedFormat),
THEN
{
    RC_Used_in_CRC = 0x0000
    IF (Multi-Cast),
    THEN
    {

```



```

// The rollover value is initialized to the local Time Stamp clock value plus an offset
// This is considered as a random number for this purpose
Initial_TS for SafetyOpenResponse = Producer_Clk_Count
Initial_Rollover_Value for SafetyOpenResponse =
    Producer_Clk_Count + Initial_RC_Offset
Last_Time_Stamp_For_Rollover = Initial_TS
TS_Rollover_Count = Initial_Rollover_Value
}
ENDIF
IF (Single-Cast AND Originator),
THEN
{
    Last_Time_Stamp_For_Rollover = Initial_TS from SafetyOpenResponse
    TS_Rollover_Count = Initial_Rollover_Value from SafetyOpenResponse
}
ENDIF
IF (Single-Cast AND Target),
THEN
{
    Last_Time_Stamp_For_Rollover = Initial_TS from SafetyOpen
    TS_Rollover_Count = Initial_Rollover_Value from SafetyOpen
}
ENDIF
}
ENDIF

////////////////////////////////////
// end, Cold start after connection establishment processing
////////////////////////////////////

```

#### 2-2.5.1.1.2 Example Safety Data Production Multi-Cast Consumer re-start Logic

The logic allows individual multi-cast consumers to be stopped and restarted while the multi-cast production continues to other consumers.

This section defines the logic needed to restart an individual consumer, but it does not represent an efficient implementation of how to detect that the individual connections has been restarted.

```

////////////////////////////////////
// Re-initialization of Multi-Cast production to individual consumers
////////////////////////////////////
// This Logic should be executed after a multicast consumer performs
// a successful open to a multi-cast producer while the production
// is in process to re-initialize production to that consumer number
////////////////////////////////////
FOR (Consumer_Num = 1 to Max_Consumer_Number),
{
    IF (Consumer_Open(Consumer_Num-1)transitions from Closed to Open)
    THEN
    {
        // Producer Dynamic Variables
        Consumer_Active_Idle[Consumer_Num-1] = Idle,
        S Connection_Fault[Consumer Num-1] = OK,
    }
}

```



```

    Producer_Rcvd_Time_Value[Consumer_Num-1] = 0x0000,
    Consumer_Time_Correction_Value[Consumer_Num-1] = 0x0000,
    Ping_Int_Since_Last_Time_Coord_Msg_Count[Consumer_Num-1] = 0x0000,
    Producer_Fault_Counter[Consumer_Num-1] = 0, //For ExtendedFormat only

    // Producer Derived Variables
    // Time_Drift_Constant, the minimum value is 1
    //(Time_Drift_Constant is not
    // saved but is used in the calculation of the
    // Connection_Correction_Constant below.)
    Time_Drift_Constant =
    Roundup((Timeout_Multiplier.PI [Consumer_Num-1] +1) * EPI *
            Ping_Interval_EPI_Multiplier / 320000),
    // Connection_Correction_Constant
    Connection_Correction_Constant[Consumer_Num-1] =
    Time_Drift_Constant + 1 - Time_Coord_Msg_Min_Multiplier [Consumer_Num-1],
    // Time_Coord_Response_EPI_Limit
    Time_Coord_Response_EPI_Limit[Consumer_Num-1] = Roundup((5000000 +
            (Time_Coord_Msg_Min_Multiplier[Consumer_Num-1]*128) +
            (EPI * ( Consumer_Num - 1))) / EPI),

    // Time_Coord_Response_EPI_Limit has a maximum value of 1000
    IF (Time_Coord_Response_EPI_Limit[Consumer_Num-1] > 1000),
    THEN
    {
        Time_Coord_Response_EPI_Limit[Consumer_Num-1] = 1000,
    }
    ENDIF
}
ENDIF
}
ENDFOR
IF (ExtendedFormat),
    THEN
    {
        Initial_TS for SafetyOpenResponse = Last_Time_Stamp_For_Rollover
        Initial_Rollover_Value for SafetyOpenResponse = TS_Rollover_Count
    }
    ENDIF

```

### 2-2.5.1.1.1.3 Example Combined Data Production

The following is the common required logic of the SafetyValidatorClient for all (see section 2-1.6) of safety connections.

```

////////////////////////////////////
// start of EPI Safety Data production processing
////////////////////////////////////
// This logic assumes that the application has passed the Safety
// Data and Application_Run_Idle to the Safety Validator.
//
////////////////////////////////////

```



```
// Capture the time to be used for the time stamp
Producer_Safe_Data_TS = Producer_Clk_Count,

// Check if it is time to increment the Ping Count
Ping_Interval_EPI_Count = Ping_Interval_EPI_Count + 1,

IF (Ping_Interval_EPI_Count >= Ping_Interval_EPI_Multiplier),
THEN
{
    Increment Mode_Byte.Ping_Count,
    Ping_Interval_EPI_Count = 0,
}
ENDIF

// Set the remainder of the Mode_Byte bits
Mode_Byte.Run_Idle = Application_Run_Idle,
Mode_Byte.TBD_Bit = 0,
Mode_Byte.TBD_2_Bit = 0,

// Execute the safety producer, connection type specific functions
IF (Connection_Type == multi-cast),
THEN
{
    multi_cast_producer_function(),
}
ELSE
{
    single_cast_producer_function(),
}
ENDIF

// set Mode_Byte redundant bits (2:4) to the correct values
Mode_Byte = (Mode_Byte AND 0xe3) OR ((Mode_Byte >> 3) AND 0x1c XOR 0x14),
Calculate CRC(s) based on the format used
Trigger the sending of the Data Message,

// If on DeviceNet and Multi-cast, send the Time Correction
// Message if it is time.
IF ((Connection_Type == DeviceNet) AND
    (Send_Time_Correction_Message == 1)),
THEN
{
    Trigger the sending of the Time Correction Message,
    Send_Time_Correction_Message = 0,
}
ENDIF
////////////////////////////////////
// end of safety data production processing
////////////////////////////////////

////////////////////////////////////
// Start of single-cast production function
////////////////////////////////////
```



```

single_cast_producer_function()
// Check if a Time Coordination message has been received in
// the allotted time.
IF (Ping_Interval_EPI_Count == 8),
THEN
{
    Increment Ping_Int_Since_Last_Time_Coord_Msg_Count[0],
    IF (Ping_Int_Since_Last_Time_Coord_Msg_Count[0]
    >= (Timeout_Multiplier.PI [0] + 2)),
    THEN
    {
        S Connection_Fault[0]=Faulted,
    }
    ENDIF
}
ENDIF

//Hold Data_Time_Stamp = 0 until time coordination msg received
IF(Consumer_Active_Idle[0] == Idle),
THEN
{
    Mode_Byte.Run_Idle = Idle,
    Time_Stamp_Section.Data_Time_Stamp = 0x0000,
    IF (ExtendedFormat),
    THEN
    {
        // Set seed value to 0 for time stamp equal to 0
        RC_Used_in_CRC = 0x0000
    }
    ENDIF
}
ELSE
{
    Time_Stamp_Section.Data_Time_Stamp = Producer_Safe_Data_TS +
    Consumer_Time_Correction_Value[0],
    IF (ExtendedFormat),
    THEN
    {
        // check for rollover
        IF (unsigned compare(Time_Stamp_Section.Data_Time_Stamp <
        Last_Time_Stamp_For_Rollover))

        THEN
        {
            TS_Rollover_Count = TS_Rollover_Count + 1
        }
        ENDIF
        // set the seed value
        RC_Used_in_CRC = TS_Rollover_Count
        // save the time stamp
        Last_Time_Stamp_For_Rollover = Time_Stamp_Section.Data_Time_Stamp
    }
    ENDIF
}
ENDIF

END single_cast_producer_function()

```



```

////////////////////////////////////
// end of single-cast processing
////////////////////////////////////

////////////////////////////////////
// Start of multi-cast production function
////////////////////////////////////
multi_cast_producer_function()
// Check if its time to start sending Time Correction messages
// Time Correction messages are sent starting at the 8th EPI production
// within the Ping interval
IF (Ping_Interval_EPI_Count == 8),
THEN
{
    RR_Con_Num_Index_Pntr = 0,
}
ENDIF

// Check if a Time Coordination message has been received in
// the allotted time.
IF (RR_Con_Num_Index_Pntr < Max_Consumer_Number),
THEN
{
    Increment Ping_Int_Since_Last_Time_Coord_Msg_Count[RR_Con_Num_Index_Pntr],
    IF (Ping_Int_Since_Last_Time_Coord_Msg_Count[RR_Con_Num_Index_Pntr]
    >= (Timeout_Multiplier.PI [RR_Con_Num_Index_Pntr] + 2)),
THEN
{
    Set S_Connection_Fault[RR_Con_Num_Index_Pntr]=Faulted,
}
ENDIF
ENDIF

// Check if its time to send an active non-errored Time Correction
// message
IF ((RR_Con_Num_Index_Pntr < Max_Consumer_Number) AND
(Consumer_Active_Idle[RR_Con_Num_Index_Pntr] == Active)), AND
(Consumer_Open[RR_Con_Num_Index_Pntr] == Open) AND
(S_Connection_Fault[RR_Con_Num_Index_Pntr] == OK)
THEN
{
    Send_Time_Correction_Message = 1,
    Mcast_Byte.Consumer_# = RR_Con_Num_Index_Pntr + 1,
    Mcast_Byte.Multi_Cast_Active_Idle =
        Consumer_Active_Idle[RR_Con_Num_Index_Pntr],
    Time_Correction_Section.Time_Correction_Value =
    Consumer_Time_Correction_Value[RR_Con_Num_Index_Pntr],
    Set Mcast_Byte.Parity_Even to the correct value,
    MCast_Byte_2 = ((MCast_Byte XOR 0xFF) AND 0x55) OR
        (MCast_Byte AND 0xAA)
    Calculate CRC of time correction message,
}
ELSE
{

```



```

        //load null TIME_CORRECTION Section,
        Mcast_Byte.Consumer_# = 0,
        Mcast_Byte.Multi_Cast_Active_Idle = Idle,
        Time_Correction_Section.Time_Correction_Value = 0,
        Set Mcast_Byte.Parity_Even to the correct value,
        //Mcast_Byte_2 not included in ExtendedFormat
        MCast_Byte_2 = ((Mcast_Byte XOR 0xFF) AND 0x55) OR
        (Mcast_Byte AND 0xAA)
        Calculate CRC of time correction section based on format used,,
    }
ENDIF

// If Time Correction messages are being sent, increment to the
// next consumer
IF (RR_Con_Num_Index_Pntr < Max_Consumer_Number),
THEN
{
    RR_Con_Num_Index_Pntr = RR_Con_Num_Index_Pntr + 1,
}
ENDIF

// Set the time stamp value
Time_Stamp_Section.Data_Time_Stamp = Producer_Safe_Data_TS,
IF (ExtendedFormat),
THEN
{
    // check for rollover
    IF (unsigned compare(Time_Stamp_Section.Data_Time_Stamp < Last_Time_Stamp_For_Rollover))
    THEN
    {
        TS_Rollover_Count = TS_Rollover_Count + 1
    }
    ENDIF
    // set the seed value
    RC_Used_in_CRC = TS_Rollover_Count
    // save the time stamp
    Last_Time_Stamp_For_Rollover = Time_Stamp_Section.Data_Time_Stamp
}
ENDIF
}
END multi_cast_producer_function()
////////////////////////////////////
// end of multi-cast processing
////////////////////////////////////

```

#### 2-2.5.1.1.1.4 Example Time Coordination Message Reception Logic

The SafetyValidatorClient must complete the following steps for each time coordination message received. For single-cast and multi-cast safety connections, a Consumer\_Time\_Value is received in the time coordination message.

FRS333 Producers shall allow Time Coordination responses to arrive during the Ping\_interval, or Ping\_Interval + 1.



```

////////////////////////////////////
//Time Coordination message reception processing
////////////////////////////////////
// Consumer_Num equals 1 for single-cast
// Consumer_Num equals the consumer the message
// was received from for multi-cast
////////////////////////////////////
// This Logic should only be executed If Ack_Byte.Ping_Response
// of the message is equal to 1 AND Consumer_Open[Consumer_Num-1]
// is equal to Open. If Ack_Byte.Ping_Response is equal to 0 OR
// Consumer_Open[Consumer_Num-1] is equal to Closed, this
// message should be ignored.
////////////////////////////////////
// reject Time Coordination packets if CONSUMER_TIMESTAMP has previously been captured AND
// this is not the first time coordination message from this consumer
IF ( (Consumer_Time_Value[Consumer_Num-1] ==
Time_Coordination_Section.Consumer_Time_Value) AND
(Consumer_Active_Idle[Consumer_Num-1] == Active) )
THEN
{
    // the only time this would be the case is if that Time Coordination was already received
    Abort the processing of this packet. Do Not close the connection.
}
ENDIF

// Capture the received time
Producer_Rcvd_Time_Value[Consumer_Num-1] = Producer_Clk_Count,

// Perform the integrity checks
Check the CRC of the Time Coordination Section, based on format used,
Ack_Byte_Error = false,
IF ((Ack_Byte parity incorrect)
THEN
{
    Ack_Byte_Error = true,
}
ENDIF
IF (BaseFormat)
THEN
{
    //check Ack_Byte_2 if BaseFormat
    IF ((Ack_Byte parity incorrect) OR (Ack_Byte_2 !=
(((Ack_Byte XOR 0xFF) AND 0x55) OR (Ack_Byte AND 0xAA))))
    THEN
    {
        Ack_Byte_Error = true,
    }
    ELSE
    {
        Ack_Byte_Error = false,
    }
    ENDIF
ENDIF
ENDIF

```



```

IF ((CRC_Error based on format used) OR (Ack_Byte_Error = true) OR
// Ensure that the Time Coordination message returned
// with the same ping interval or the next ping interval.
// The next ping interval is allowed for the case that a
// multi-cast consumer connects to an existing producer.
// If the new consumer receives its first message near the end
// of the ping interval, the time coordination may arrive back
// at the producer during the next ping interval.
  ((Ack_Byte.Ping_Count_Reply != Ping_Count) AND
   (Ack_Byte.Ping_Count_Reply != Ping_Count-1)) OR
// Ensure that the Time Coordination message returned
// within the approximatley 5 second limit
((Consumer_Active_Idle[Consumer_Num-1] == Active) AND
 (Ping_Interval_EPI_Count >
  Time_Coord_Response_EPI_Limit[Consumer_Num-1])),
THEN
{
  IF (ExtendedFormat),
  THEN
  {
    //decide whether to close the connection or not
    Increment Producer_Fault_Counter[Consumer_Num-1]
    IF (Producer_Fault_Counter[Consumer_Num-1] >= Max_Fault_Number)
    THEN
    {
      S_Connection_Fault[Consumer_Num-1] = Faulted,
      Abort the processing of this packet. Close the connection.
    }
    ELSE
    {
      Abort the processing of this packet. Do Not close the connection.
    }
    ENDIF
  }
  ELSE
  {
    S_Connection_Fault[Consumer_Num-1] = Faulted,
    Abort the processing of this packet. Close the connection.
  }
  ENDIF
}
]
ENDIF

// at this point we know we have a good, non-redundant time coordination packet

// Capture the received CONSUMER time
Consumer_Time_Value[Consumer_Num-1] =
  Time_Coordination_Section.Consumer_Time_Value;

// Determine the worst case Consumer Time Correction_Value based on the
// received Time Coordination section
Worst_Case_Consumer_Time_Correction_Value[Consumer_Num-1] =
  Time_Coordination_Section.Consumer_Time_Value
  - Producer_Rcvd_Time_Value[Consumer_Num-1]
  - Connection_Correction_Constant[Consumer_Num-1],

```



```

// Determine the worst case time drift since the last
// Time Coordination information was received. This is equal to the
// number of Ping Intervals since the last Time Coordination message
// times the Time Drift per Ping Interval plus 1 for the asynchronous
// clocks. This value will not be used until the second
// Time_Coordination message is received.

Time_Drift_Since_Last_Time_Coord =
({Ping_Int_Since_Last_Time_Coord_Msg_Count[Consumer_Num-1]+ 1}
 * Time_Drift_Per_Ping_Interval) + 1,

// If this Time Coordination Transport delay is greater then the
// last Time_Coordination Transport delay by more than the appropriate
// number of Time_Drift_Per_Ping_Intervals, then, use the previous
// Correction value minus the appropriate number of
// Time_Drift_Per_Ping_Intervals. This action is not taken until the
// second Time_Coordination message is received.
// The math for this check should be 16 bit or greater
IF ((Consumer_Active_Idle[Consumer_Num-1] == Active) AND
((Consumer_Time_Correction_Value[Consumer_Num-1]-
Worst_Case_Consumer_Time_Correction_Value[Consumer_Num-1] -
Time_Drift_Since_Last_Time_Coord) AND 0x8000 == 0)),
THEN
{
Consumer_Time_Correction_Value[Consumer_Num-1] =
Consumer_Time_Correction_Value[Consumer_Num-1] -
Time_Drift_Since_Last_Time_Coord,
}
ELSE
{
Consumer_Time_Correction_Value[Consumer_Num-1] =
Worst_Case_Consumer_Time_Correction_Value[Consumer_Num-1],
}
ENDIF

// Set the flag indicating a Time_Coordination message has been received
IF (S_Connection_Fault[Consumer_Num-1] = OK)
THEN
{
Consumer_Active_Idle[Consumer_Num-1] = Active,
}
ENDIF

// Reset the Time_Coordination message timer
Ping_Int_Since_Last_Time_Coord_Msg_Count[Consumer_Num-1] = 0,
////////////////////////////////////
// end producer time coordination information reception processing
////////////////////////////////////

```

## 2-2.6 SafetyValidatorServer Function Definition

The SafetyValidatorServer function is the safety layer that augments a standard CIP connection to transmit SIL 3 data. It coordinates the reception of safety data with a peer instance of the SafetyValidatorClient class and delivers this data to the consuming safety application.



## 2-2.6.1 Safety Consumption

This section describes the consumption of data and the interfaces between the application, the SafetyValidatorServer and the underlying link layer.

There are two versions of safety data consumer monitoring. They are:

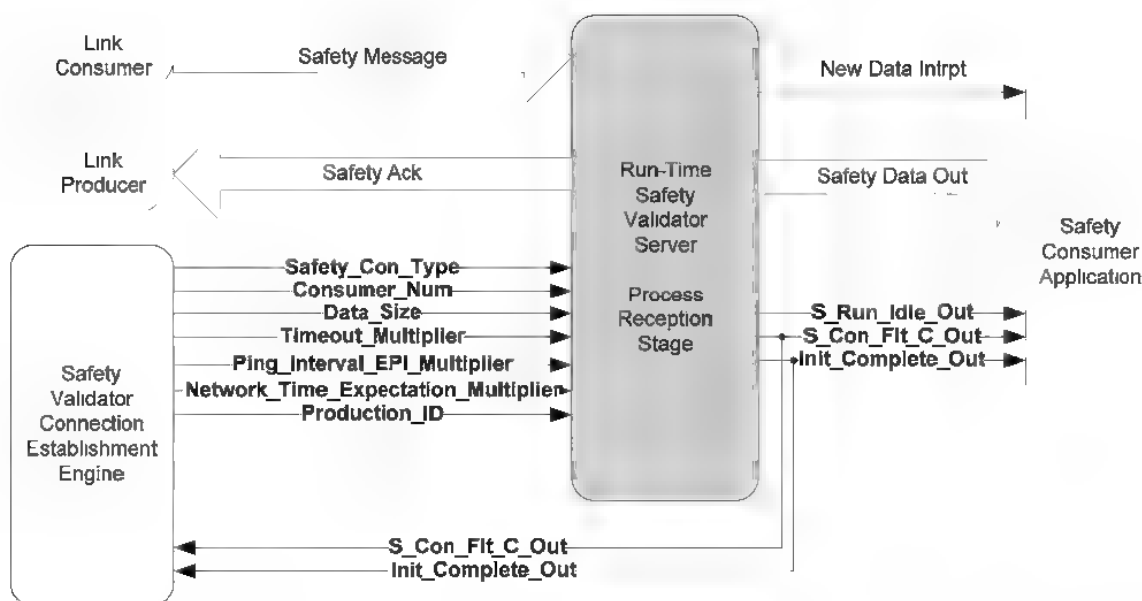
- **SafetyValidatorServer - Link Triggered.**  
This option is appropriate for consuming applications that are synchronized to the reception of the data, or consuming applications that present a continuous delay from the reception of data.
- **SafetyValidatorServer - Application Triggered.**  
This option is appropriate for consuming applications that are periodic in nature and not synchronized to the reception of the data.

In both versions, FRS127 the SafetyValidatorServer shall perform the following aspects of the safety data monitoring:

- Time coordination with the producer
- Time stamp section integrity checking
- Data integrity checking
- Network time expectation checking

### 2-2.6.1.1 SafetyValidatorServer - Link Triggered

Figure 2-2.7 Consumer Safety Data Monitoring



FRS128 The Link Triggered SafetyValidatorServer shall perform all aspects of the safety protocol on each message received.



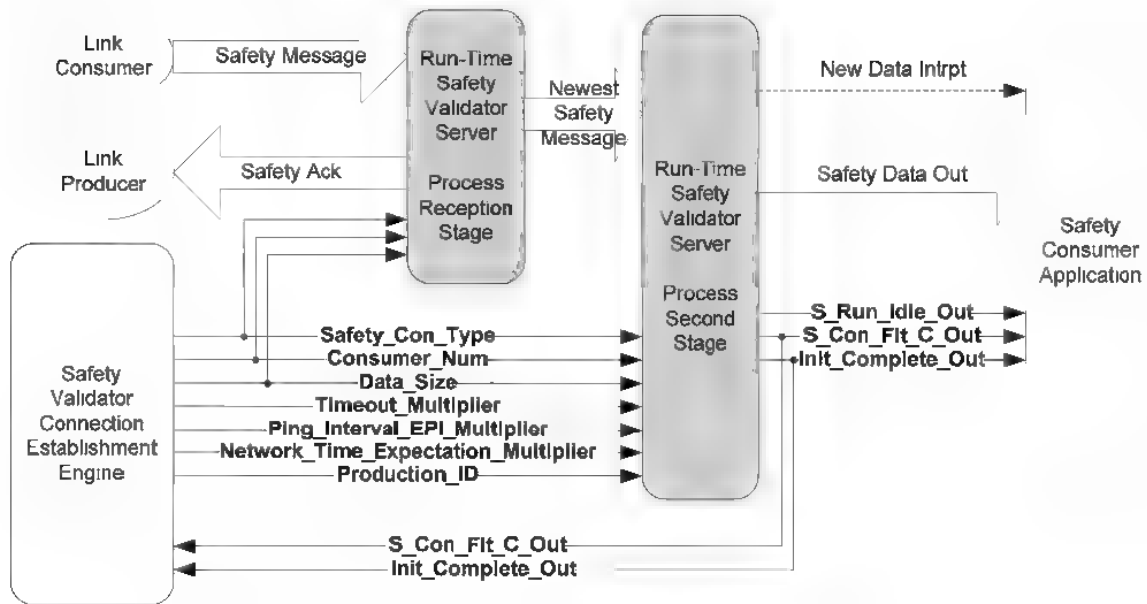
FRS129 For multi-cast, the Link Triggered SafetyValidatorServer shall also perform all aspects of the safety protocol on each Time\_Correction section received.

FRS130 The SafetyValidatorServer shall insure that the safety data presented to the consuming application has been crosschecked and that the network time expectation is being met.

FRS131 The Link Triggered SafetyValidatorServer shall set the new data flag each time data is ready for consumption.

### 2-2.6.1.2 SafetyValidatorServer - Application Triggered

Figure 2-2.8 SafetyValidatorServer - Application Triggered



The Application Triggered SafetyValidatorServer does not fully process each message.

FRS132 The process reception stage of the application triggered SafetyValidatorServer shall check the ping count for all incoming data messages.

FRS133 For multi-cast consumption in application triggered SafetyValidatorServers, the latest Time\_Correction section received shall be forwarded to the consuming application.

FRS134 The consuming safety application shall request new data from the SafetyValidatorServer in order to cause the SafetyValidatorServer to complete processing of a message.

For periodic applications that are not synchronized to the reception of the data, this reduces the processing burden, since the data section is only crosschecked (in Process Second Stage in Figure 2-2.8) for the safety data messages that are used. This is particularly useful for the case where the network EPIs are at a faster rate than the consuming application's periodic rate.

FRS135 The consuming application shall ensure that it gets new data at least once every 2 seconds to ensure that the time stamp does not rollover.



### 2-2.6.2 Safety Data Reception Logic

The safety data reception processing consists of:

- Ping\_Count check
- Time Stamp section checking
- Data integrity checking
- Time stamp checking
- Deriving Run Idle and fault status

The definitions of the variables used in the safety data reception logic can be found in section 2-4.6

The following table shows the safety data processing steps that are performed by the Link Triggered SafetyValidatorServer.

**Table 2-2.1 Data Reception - Link Triggered**

	Single-Cast	Multi-Cast
Check Ping_Response	No	No
Check Ping_Count	Yes	Yes
Check Time stamp section CRC	Yes	Yes
Check Run_Idle	Yes	Yes
Check TBD_Bit	Yes	Yes
Check Data CRCs	Yes	Yes
Check Data Actual/Complement	Yes	Yes
Derive Data_Time Stamp	Yes	Yes

The following table shows the safety data processing steps that shall be performed by the Link Triggered SafetyValidatorServer for the reception of each Time Correction section.

**Table 2-2.2 Time Correction Reception - Link Triggered**

	Multi-Cast
Check MCast_Byte and MCast_Byte 2	Yes, if consumer # match
Check Time Correction CRC	Yes, if consumer # match
Get Time Correction_Value	Yes, if consumer # match
Check Multi_Cast Active Idle	Yes, if consumer # match

The following table shows the safety data processing steps that are performed on each message reception by the application triggered SafetyValidatorServer.



**Table 2-2.3 Data Reception - Application Triggered**

	Single-Cast	Multi-Cast
Check Ping_Response	No	No
Check Ping_Count	Yes	Yes
Check Time stamp section CRC	No	No
Check Run_Idle	No	No
Check TBD_Bit	No	No
Check Data CRCs	No	No
Check Data Actual/Complement	No	No
Derive Data_Time_Stamp	No	No

The following table shows the safety data processing steps that shall be performed on each Time\_Correction section reception by the application triggered SafetyValidatorServers.

**Table 2-2.4 Time\_Correction Reception - Application Triggered**

	Multi-Cast
Check MCast_Byte and MCast_Byte_2	Forward message if consumer_# match
Check Time_Correction CRC	No
Get Time_Correction_Value	no
Check Multi_Cast_Active_Idle	No

The following table shows the safety data processing steps that are performed by the application triggered SafetyValidatorServer when safety data is requested.

**Table 2-2.5 Consuming Application – Safety Data Monitoring**

	Single-Cast	Multi-Cast
Check Ping_Response		NA
Check Ping_Count	NA	NA
Check Time stamp section CRC	Yes	Yes
Check MCast_Byte and MCast_Byte_2	NA	Yes (from last time correction)
Check Time_Correction CRC	NA	Yes (from last time correction)
Get Time_Correction_Value	NA	Yes (from last time correction)
Check Multi_Cast_Active_Idle	NA	Yes (from last time correction)
Check Run_Idle	Yes	Yes
Check TBD_Bit	Yes	Yes
Check Data CRCs	Yes	Yes
Check Data Actual/Complement	Yes	Yes
Derive Data_Time_Stamp	Yes	Yes

### **2-2.6.2.1 Ping Count Checking**

FRS142 The SafetyValidatorServer shall perform Ping\_Count checking on every message . Ping\_Count checking checks the ping count to see if it has changed.

FRS332 If a SafetyValidatorServer detects that the ping count has changed, it shall produce a time coordination message within that Ping Interval or within 5 seconds, whichever is less.



Ping\_Count checking errors should not be treated as dangerous failures. The time stamp section CRC does not need to be checked to perform the Ping\_Count checking.

### **2-2.6.2.2 Data and Network Time Expectation Checking Maximum Interval**

FRS145 The data integrity shall be checked at least once every 2 seconds.

FRS146 The Time Stamp and Network Time Expectation (NTE) shall be checked at least once every 2 seconds

### **2-2.6.2.3 Example Cold Start Initialization**

```
////////////////////////////////////
// Cold start after connection establishment processing
////////////////////////////////////
// This Logic should be executed at the transition of the
// consuming connection from closed to open.
////////////////////////////////////

S_Con_Flt_C_Out = OK,
Init_Complete_Out = 0,
S_Run_Idle_Out = Idle,

Last_Ping_Count = 3,
Time_Coordination_Count_Down = 0x0000,

Corrected_Data_Time_Stamp = 0x0000,
Last_Data_Time_Stamp = 0x0000,
Last_Rcvd_Multi_Cast_Active_Idle = Idle,
Last_Rcvd_Time_Correction_Value = 0x0000,
Time_Correction_Ping_Interval_Count = 0x0000,
Time_Correction_Received_Flag = 0,
Data_Age = 0x0000,
Max_Data_Age = 0x0000,
IF (ExtendedFormat),
THEN
{
    Consumer_Fault_Counter = 0
    RC_Used_in_CRC = 0x0000
    IF (Multi-cast),
    THEN
    {
        Last_Time_Stamp_For_Rollover = Initial_TS from SafetyOpenResponse
        TS_Rollover_Count = Initial_Rollover_Value from SafetyOpenResponse
    }
    ENDIF
    IF (Single-cast),
    THEN
    {
        IF (Originator)
        THEN
        {
            Initial_TS for SafetyOpen = Consumer_Clk_Count
            Initial_Rollover_Value for SafetyOpen = Consumer_Clk_Count
        }
    }
}
```



```

+ Initial_RC_Offset
}
IF (Target)
THEN
{
    Initial_TS for SafetyOpenResponse = Consumer_Clk_Count
    Initial_Rollover_Value for SafetyOpenResponse = Consumer_Clk_Count
+ Initial_RC_Offset
}
Initial_RC_Offset = Initial_RC_Offset + 1
Last_Time_Stamp_For_Rollover = Initial_TS
TS_Rollover_Count = Initial_Rollover_Value
}
ENDIF

}
ENDIF
////////////////////////////////////
// end, Cold start after connection establishment processing
////////////////////////////////////

```

### 2-2.6.3 SafetyValidatorServer - Link Triggered Logic

The following three sections describe the consumer safety data reception logic for the SafetyValidatorServer - Link Triggered option. The term Link Triggered means that all the logic is executed on every reception.

The first section describes the combined logic associated with the Safety data packet reception for the single-cast and multi-cast safety connection types.

The third section describes the logic associated with the Time Correction Message reception for the multi-cast safety connection type.

This logic should only be executed if the connection is open with no errors.

#### 2-2.6.3.1 Example Combined Reception Logic - Link Triggered Logic

The following is the required SafetyValidatorServer reception logic for the safety data packet reception of the single-cast or multi-cast safety connection type, for the SafetyValidatorServer – - Link Triggered Logic.

Multi-cast connections shall send a Time Coordination Message (Consumer\_Num – 1) receptions after the ping count change.

FRS148 Multicast-consumers connecting to an existing producer shall send the Time\_Coordination message immediately upon the first valid reception.

```

////////////////////////////////////
// Safety Data Consumption - link triggered logic
////////////////////////////////////

// If the extended format is being used, we must determine a rollover value to

```



```

// be used when calculating CRCs before we attempt to validate the message
// (since the rollover is part of CRC values in the message).
IF (ExtendedFormat),
THEN
{
    // For single-cast, a time_stamp other than 0 or a Run
    // indication, indicate that the Time_Coordination_Message has been
    // received by the Client and the corrected time stamp is being used.
    IF ((SingleCast) AND (Init_Complete_Out == 0) AND
        (Time_Stamp_Section.Data_Time_Stamp == 0) AND
        (Mode_Byte.Run_Idle == IDLE)),
    THEN
    {
        RC_Used_in_CRC = 0x0000,
    }
    ELSE
    {
        RC_Used_in_CRC = TS_Rollover_Count,

        // check for time stamp rollover
        IF (unsigned compare(Time_Stamp_Section.Data_Time_Stamp <
            Last_Time_Stamp_For_Rollover))
        THEN
        {
            RC_Used_in_CRC = RC_Used_in_CRC + 1,
        }
        ENDIF
    }
    ENDIF
}
ENDIF

Temp_Fault_Flag = OK,

// check for data integrity faults
IF ((CRC_Error_based_on_format_used)
    OR (Complement_Data_Section_CRC != OK)
    OR (Mode_Byte.Run_Idle != not Mode_Byte.N_Run_Idle)
    OR (Mode_Byte.TBD_2_Bit !=
        Mode_Byte.TBD_2_Bit_Copy)
    OR (Mode_Byte.TBD_Bit != not Mode_Byte.N_TBD_Bit)
    OR (Actual_Data != not Complement_Data) [for > 2 bytes only]
    ),
THEN
{
    Temp_Fault_Flag = Faulted,
}
ENDIF

// Set up Time_Stamp_Delta, to be used to check for out of sequence
// messages Time_Stamp_Delta is not calculated until the EPI
// reception after Init_Complete_Out is set to 1.
IF(Init_Complete_Out == 0),
THEN
{
    Time_Stamp_Delta = 1,

```



```
]
ELSE
{
    Time_Stamp_Delta = Time_Stamp_Section.Data_Time_Stamp -
    Last_Data_Time_Stamp,
}
ENDIF

// Take a look at the time stamp delta and determine if it indicates
// the message was received in the right order.
IF ((Time_Stamp_Delta < 0) OR
    (Time_Stamp_Delta > Network_Time_Expectation_Multiplier)),
THEN
{
    Temp_Fault_Flag = Faulted,
}
ENDIF

// If the extended format is being used, we can drop some packets with
// data integrity/message validation or out of order errors.
IF ((ExtendedFormat) AND (Temp_Fault_Flag == Faulted)),
THEN
{
    increment Consumer_Fault_Counter

    // Calculate updated data age for the data in the last good packet
    IF (Init_Complete_Out)
    THEN
    {
        Data_Age = Consumer_Clk_Count - Last_Corrected_Data_Time_Stamp,
    }
    ENDIF

    // Calculate the maximum worst case age
    IF (Data_Age > Max_Data_Age)
    THEN
    {
        Max_Data_Age = Data_Age,
    }
    ENDIF

    // Is it OK to ignore this error?
    // It is OK if the consumer fault counter is not greater than the
    // maximum faults allowed and the updated data age from the
    // last good packet still meets our data age criteria
    // (less than the time expectation multiplier)
    IF (
        (Consumer_Fault_Counter < Max_Fault_Number) AND
        (Data_Age <= Network_Time_Expectation_Multiplier)
    )
    THEN
    {
        Abort the processing of this packet. Do Not close the connection.
        Do not use the Data.
        Do not update Last_Corrected_Data_Time_Stamp or Last_Data_Time_Stamp
    }
}
```



```
ELSE
{
    S_Con_Flt_C_Out = Faulted,
    Abort the processing of this packet. Close the connection.
}
ENDIF
}
ENDIF

// execute a function that checks for ping count changes and
// determines if it is time to produce a time coordination message
ping_count check in consumer function(),

Init_Complete_Out_Temp = 0,

// execute the safety connection type specific functions
IF (Connection_Type == Multi-cast),
THEN
{
    multi_cast_consumer_function(),
}
ELSE
{
    single_cast_consumer_function(),
}
ENDIF

// If we are using the ExtendedFormat, at this point we have a properly formatted
// message that was received in the right order. With the extended format
// update the rollover count and last timestamp for rollover.
IF (ExtendedFormat),
THEN
{
    // Only update the rollover count if this is a multicast connection
    // or if this is a singlecast connection that is out of the initialization
    // phase.
    IF ((Connection_Type == Multi-cast) OR
        (Init_Complete_Out_Temp == 1)),
    THEN
    {
        // Check the timestamp value in the message to determine if a
        // rollover has occurred, and if so, update the rollover count
        // accordingly.
        IF (unsigned compare(Time_Stamp_Section.Data_Time_Stamp <
            Last_Time_Stamp_For_Rollover))
        THEN
        {
            TS_Rollover_Count = TS_Rollover_Count + 1,
        }
        ENDIF

        Last_Time_Stamp_For_Rollover = Time_Stamp_Section.Data_Time_Stamp,
    }
    ENDIF

    // Save the corrected data timestamp so we can recalculate a data age
```



```

    // if the next packet has errors.
    Last_Corrected_Data_Time_Stamp = Corrected_Data_Time_Stamp,
}
ENDIF

// Save Time stamp for next reception sequence check
Last_Data_Time_Stamp = Time_Stamp_Section.Data_Time_Stamp,

// Calculate the worst case age
IF (Init_Complete_Out_Temp == 1),
THEN
{
    Data_Age = Consumer_Clk_Count - Corrected_Data_Time_Stamp,
}
ENDIF

// Calculate the maximum worst case age
IF (Data_Age > Max_Data_Age)
THEN
{
    Max_Data_Age = Data_Age,
}
ENDIF

// Check for Age, Integrity, and out of sequence faults
IF (( unsigned int_16( Data_Age > Network_Time_Expectation_Multiplier)
    AND (Init_Complete_Out_Temp == 1))
    OR (Temp_Fault_Flag == Faulted))
THEN
{
    // The connection will be closed, the data should not be used
    S_Con_Flt_C_Out_Temp = Faulted,
}
ENDIF

////////////////////////////////////
// The following information is passed on to the consuming application
// with block integrity: Safety Data,Init Complete Out,S Run Idle Out
// and S_Con_Flt_C_Out.
// The time between the checking of the age of the data and passing
// the data off to the consuming application must be protected under
// the time stamp check or the consuming application time checks.
////////////////////////////////////
Make Safety Data Available to Consuming Application
Init_Complete_Out = Init_Complete_Out_Temp,
S_Run_Idle_Out = S_Run_Idle_Out_Temp,
S_Con_Flt_C_Out = S_Con_Flt_C_Out_Temp,
////////////////////////////////////
// end Safety Data Consumption - link triggered logic
////////////////////////////////////

```



```
////////////////////////////////////
// Start of single-cast consumer function
////////////////////////////////////
single_cast_consumer_function()
// For single-cast multicast, no time stamp correction is required
Corrected_Data_Time_Stamp = Time_Stamp_Section.Data_Time_Stamp,

// Determine the Init_Complete_Out state to be passed on to the
// consuming application.
// For single-cast or bi-dir, a time_stamp other than 0 or a Run
// indication, indicate that the Time_Coordination_Message has been
// received by the Client and the corrected time stamp is being used.
IF ((Init_Complete_Out == 1) OR
    (Corrected_Data_Time_Stamp != 0) OR
    (Mode_Byte.Run_Idle == Run)),
THEN
{
    Init_Complete_Out_Temp = 1,
}
ENDIF

// Determine the Run_Idle indication to be passed on to the consuming
// application.
IF (((Mode_Byte.Run_Idle == Idle) OR (Init_Complete_Out_Temp == 0))
THEN
{
    S_Run_Idle_Out_Temp = Idle,
}
ELSE
{
    S_Run_Idle_Out_Temp = Run,
}
ENDIF

END single_cast_consumer_function()
////////////////////////////////////
// end of single-cast processing
////////////////////////////////////

////////////////////////////////////
// Start of multi-cast consumer function
////////////////////////////////////
multi_cast_consumer_function()
// If multi-cast, check that a time correction message has been
// received within the last Timeout_Multiplier.PI + 1 Ping Intervals
IF (Time_Correction_Ping_Interval_Count > (Timeout_Multiplier.PI + 1))
THEN
{
    Temp_Fault_Flag = Faulted,
}
ENDIF

// If multicast, add correction value to time stamp
Corrected_Data_Time_Stamp = Time_Stamp_Section.Data_Time_Stamp +
Last_Rcvd_Time_Correction_Value
```



```

// Determine the Init_Complete_Out state to be passed on to the
// consuming application.
// For multi-cast, the reception of a Time_Coorection Message, indicate
// that data may be used.
IF (Time_Correction_Received_Flag == 1)
THEN
{
    Init_Complete_Out_Temp = 1,
}
ENDIF
// Determine the Run_Idle indication to be passed on to the consuming
// application.
IF ((Mode_Byte.Run_Idle == Idle) OR
    (Init_Complete_Out_Temp == 0) OR
    (Time_Correction_Received_Flag == 0) OR //used only on multi-cast
    (Last_Rcvd_Multi_Cast_Active_Idle==Idle))//used only on multi-cast
THEN
{
    S_Run_Idle_Out_Temp = Idle,
}
ELSE
{
    S_Run_Idle_Out_Temp = Run,
}
ENDIF
end multi_cast_consumer_function()
////////////////////////////////////
// end of multi-cast processing
////////////////////////////////////

////////////////////////////////////
// Start of ping count check function
////////////////////////////////////
ping_count_check_in_consumer_function()

// Check if its time to send a Time Coordination Message
// Multi-cast connections send a Time Coordination Message
// (Consumer_Num - 1) receptions after the ping count change.
IF (received Mode_Byte.Ping_Count != Last_Ping_Count),
THEN
{
    Time_Coordination_Count_Down = Consumer_Num,
    Increment Time_Correction_Ping_Interval_Count,
}
ENDIF

// For multicast-consumers connecting to an existing producer, send
// the Time_Coordination message immediately upon the 1st reception.
IF (It is the first data reception),
THEN
{
    Produce Time_Coordination_Message,
    Time_Coordination_Count_Down = 0,
}
ENDIF

```



```
// Check if a ping count change has been detected and the Time
// Coordination message has not been sent yet.
IF (Time_Coordination_Count_Down > 0),
THEN
{
    decrement Time_Coordination_Count_Down,

    // Send the Time_Coordination_message if the decremented
    // Time_Coordination_Count_Down transitions to 0
    IF (Time_Coordination_Count_Down == 0),
    THEN
    {
        Produce Time_Coordination_Message,
    }
    ENDIF
}
ENDIF

// Save ping count for next reception check
Last_Ping_Count = Mode_Byte.Ping_Count,

end ping_count_check_in_consumer_function()
/////////////////////////////////////////////////////////////////
// end of ping count check function
/////////////////////////////////////////////////////////////////
```

#### 2-2.6.3.2 Example Time Correction Message Reception - Link Triggered Logic

The following is the required SafetyValidatorServer logic associated with the Time Correction Message reception for the multi-cast safety connection type, for the Consumer – Link Triggered.

```
/////////////////////////////////////////////////////////////////
// Consumer processing - Time Correction Message reception
/////////////////////////////////////////////////////////////////
// Is the Time Correction message for this Consumer ?
IF (MCast_Byte.Consumer_Num = Consumer_Num)
// Perform integrity checks.
THEN
{
    IF (Time_Coordination_message_CRC_incorrect)
    OR (MCast_Byte_parity_incorrect)
    // MCast_Byte_2 not included in ExtendedFormat
    OR {MCast_Byte_2 != (((MCast_Byte_XOR0xFF)AND0x55)
        OR(MCast_Byte_AND0xAA))}
        OR {(MCast_Byte.Multi_Cast_Active_Idle == Idle)
    AND (Last_Rcvd_Multi_Cast_Active_Idle == Active)},
    // only one transition from connection idle to active will
    // be allowed, to prevent an erroneous idle indication,
    // a transition from active to idle will be latched into
    // a fault state
    THEN
    {
        IF (ExtendedFormat)
        THEN
        {
```



```
{ increment Consumer_Fault_Counter
IF (Consumer_Fault_Counter < Max_Fault_Number)
THEN
{
    Abort the processing of this packet. Do not close the connection,
    Do not use the Data.
}
ELSE
{
    S_Con_Flt_C_Out = Faulted,
    Abort the processing of this packet. Close the connection
}
ENDIF
}
ELSE
{
    S_Con_Flt_C_Out = Faulted
}
}
ENDIF

// Save the Time_Correction_Value.
Last_Rcvd_Time_Correction_Value =
Time_Correction_Section.Time_Correction_Value,

// Save the Multi_Cast_Active_Idle bit.
Last_Rcvd_Multi_Cast_Active_Idle =
MCast_Byte.Multi_Cast_Active_Idle,

// Set the Time_Correction_Received_Flag.
Time_Correction_Received_Flag = 1,

// Reset the Time Correction timeout counter
Time_Correction_Ping_Interval_Count = 0,
ENDIF
////////////////////
// end - Time Correction Message reception
////////////////////
```

#### **2-2.6.4 SafetyValidatorServer - Application Triggered Logic**

For SafetyValidatorServer - Application Triggered, the same logic described above must be performed by the combination of the SafetyValidatorServer, but only when requested by the consuming application. Application Triggered implementations only process the safety packets when needed by the application. This implementation strategy is useful for applications that usually run slower than a typical EPI rate. The functions can be divided as shown in Table 2-2.3 and Table 2-2.5.

The protocol processing steps shown in Table 2-2.5 only need to be performed on the messages used by the consuming application.



### 2-2.6.5 Example Time Coordination Message Production Logic

The SafetyValidatorServer must perform the following steps each time a time coordination message is sent. For single-cast and multi-cast safety connections, time coordination data is sent in the time coordination message. This logic is also described in the single\_cast\_function() of the combined production logic section.

```

////////////////////////////////////
//Time Coordination message production processing
////////////////////////////////////
// Set the Ping_Count_Reply to the value captured at the ping request
Ack_Byte.Ping_Count_Reply = Mode_Byte.Ping_Count,

// Set the Ping_Response bit to 1
Ack_Byte.Ping_Response = 1,

// Set the Consumer_Time_Value to the Consumer clock value
Time_Coordination_section.Consumer_Time_Value = Consumer_Clk_Count,

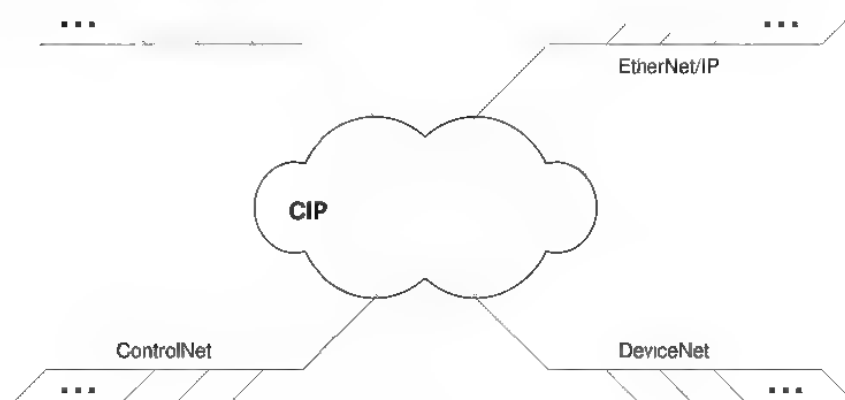
// Add data integrity measures
Set Ack_Byte.Reserved(6:4) = 0,
Set Ack_Byte.Reserved(2) = 0,
Set Ack_Byte.Parity_Even bit to the correct value,
//Ack_Byte_2 not included in ExtendedFormat
Ack_Byte_2 !=((Ack_Byte XOR 0xFF)AND 0x55)OR
(Ack_Byte AND 0xAA)),
Calculate CRC on Time Coordination message,
////////////////////////////////////
// end time coordination message send logic
////////////////////////////////////

```

## 2-3 CIP Connection Establishment

This section provides the general requirements for CIP connection establishment between safety nodes on and between DeviceNet, EtherNet/IP or SERCOS III. Figure 2-3.1 depicts a conceptual view of the highly flexible interconnection variations that are supported:

Figure 2-3.1 Conceptual Views of CIP Connections





### 2-3.1 Overview

This section does not attempt to provide a tutorial on every aspect of CIP, but rather a high-level overview of how CIP connections are established on each network and across network types, highlighting the features used and extensions required to support Safety Connections. For details on CIP connections and data formats as supported on the various network technologies, refer to their respective specifications.

FRS153 All CIP Safety connections shall be established using a Forward\_Open with a safety network segment (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet) .

There are measures for detection of both system and user errors defined. The following table shows the errors and the measures to detect these errors.

**Table 2-3.1 Connection Establishment Errors and Measures to Detect Errors**

Connection Establishment Errors	Measures to detect connection establishment errors				
	User Testing	OUNID Verification	TUNID Verification	CPCRC CRC	Configuration ID (SCID)
User misroutes connection request	X	X <sup>1</sup>	X		
System misroutes connection request			X		
Corrupted Forward Open				X	X
Corrupted or Changed Application Configuration					X
Reconfiguration from wrong originator		X			

**Note 1; Only applies if ownership has been established.**

#### 2-3.1.1 Definition of the Measures Used During Connection Establishment

**System-Wide Unique "Safety Network Number" (SNN)** Uniquely identifies a network across all networks in the safety system. FRS154 The user safety manual shall contain an advisory that states "The user should assign SNN numbers for each safety network or safety sub-net that are unique system-wide", or words to that effect. This number shall be defined during node commissioning (i.e.: set via the Propose\_TUNID/Apply\_TUNID process defined in the SafetySupervisor if the SNCT Interface is supported).

The Safety Network Number makes up part of the UNID. This value is a 6-octet value that uses the CIP DATE\_AND\_TIME format.

The DATE values of 1 thru 11,687 decimal (January 1, 1972 thru Dec 31, 2003) are past time values that are reserved for Manual Setting values of the SNN.

The SNN may be one of two types:

1. Manual Setting
2. Software Tool Generated

The Type of SNN can be determined from the DATE section.



**Table 2-3.2 SNN Date/Time Allocations**

From DATE (Decimal)	To DATE (Decimal)	From DATE (Day)	To DATE (Day)	SNN Type
0	0	Jan 1, 1972	Jan 1, 1972	Null SNN (Illegal Value)
1	1	Jan 2, 1972	Jan 2, 1972	Manual Setting - Backplane
2	2	Jan 3, 1972	Jan 3, 1972	Manual Setting - ControlNet
3	3	Jan 4, 1972	Jan 4, 1972	Reserved for future use
4	4	Jan 5, 1972	Jan 5, 1972	Manual Setting – EtherNet/IP
5	5	Jan 6, 1972	Jan 6, 1972	Manual Setting – DeviceNet
6	6	Jan 7, 1972	Jan 7, 1972	Manual Setting – SERCOS III
7	1460	Jan 7, 1972	Dec 31, 1975	Reserved for future use
1461	11,687	Jan 1, 1976	Dec 31, 2003	Vendor Specific range
11,688	65,534	Jan 1, 2004	Jun 5, 2151	Software Tool Generated
65,535	65,535	Jun 6, 2151	Jun 6, 2151	Target indication that no SNN has been set

If a software tool reads a reserved SNN Type, it shall display it as a Hex value.

The legal Range of TIME values for the Type of SNN are listed in the following table.

**Table 2-3.3 SNN Legal Range of Time Values**

SNN Type	FromTime (Hex)	To Time (Hex)	From Time (Hour/Min/Sec)	To Time (Hour/Min/Sec)
Null SNN	0x00000000	0x00000000	00d00h00m00.000s	00d00h00m00.000s
Manual Setting (DATE = 1 thru 6)	0x00000001	0x0000270F (9,999 decimal)	00d00h00m00.001s	00d00h00m09.999s
Reserved for future use	-	-	-	-
Software Tool Generated	0x00000000	0x05265BFF	00d00h00m00.000s	00d23h59m59.999s
Target indication that no SNN has been set	0xFFFFFFFF	0xFFFFFFFF	-24d20h31m23.648s	-24d20h31m23.648s

**System-Wide "Unique Node Identifier" (UNID)** – Uniquely identifies a node across all networks in the safety system. This value is made up of the Safety Network Number (SNN) and the Node Address of the device. This value is a structure consisting of the 6-byte SNN and a 4-byte Node Address. The 4-byte node address is sized to accommodate all forms of CIP network addresses.

```
struct UNID
{
    S SNN SNN;
    UDINT MACID;
};
```

MACIDs smaller than 4-bytes (i.e., DeviceNet) shall be aligned to the least significant byte in little endian order. The unused bytes shall be zero filled.



**Safety Function** - The UNID is used to uniquely identify originators and/or targets during the connection establishment process

Other uses of UNID:

**TUNID** = Target's Unique Network Identifier

**Safety Function** – Identifies the target in the SafetyOpen

**OUNID** = Originator's Unique Network Identifier

**Safety Function** – Identifies the originator in the SafetyOpen

**OCPUNID** = Output Connection Owning UNID

**Safety Function** – Identifies the owner of the output connection in the target, this shall be equal to the OUNID in normal operation

**CFUNID** = Configuration Owning UNID

**Safety Function** – Identifies the owner of the safety configuration in the target, this shall be equal to the OUNID in normal operation

**Producer Identifier (PID)** – Identifies the producer of the runtime data and Time Correction data.

FRS155 The Producer Identifier (PID) shall be used as an initial seed value in the CRCs in the runtime protocol.

FRS156 The Consumer shall obtain the Producer Identifier from either the SafetyOpen or SafetyOpen response (depending upon originator) and use the value as an initial seed in runtime checking the safety CRCs.

The value of the PID is Vendor ID + Device Serial Number + CIP Connection Serial Number.

**Safety Function** – The PID is used to ensure the correct routing of the produced/safety connection data and Time Correction data.

**Consumer Identifier (CID)** – Identifies the producer of the Time Coordination message.

FRS330 The Consumer Identifier (CID) shall be used as an initial seed value in the CRC in the Time Coordination message.

FRS331 The Producer shall obtain the Consumer Identifier from either the SafetyOpen or the SafetyOpen Response (depending upon originator) and use the value as an initial seed in runtime checking the safety CRCs .

The value of the CID is Vendor ID + Device Serial Number + CIP Connection Serial Number.



**Safety Function** – The CID is used to ensure the correct routing of the Time Coordination data.

**Connection Parameters CRC (CPCRC)** – This is a safety related CRC covering the CIP connection establishment data in the Safety Open. This CRC will also cover the additional parameters required by the safety protocol. The CRC used for the CPCRC is CRC-S4 and is defined in Appendix E.

**Safety Function** – The CPCRC is used to ensure the integrity of the connection data contained within the Safety Open.

**Safety Configuration CRC (SCCRC)** – This is a CRC that covers the device configuration data that is contained in a Safety Open. The parameters covered by the CPCRC are not covered by the SCCRC. The CRC used for the SCCRC is CRC-S4 and is defined in Appendix E.

FRS110 The Safety Configuration Data CRC (SCCRC) shall be calculated over the entire application data section that is sent to the device being configured by the tool.

FRS158 The SCCRC itself shall be included in the calculation of the CPCRC

**Safety Function** – The SCCRC is used to ensure configuration data integrity while being transferred and to ensure configuration data integrity in storage.

**Safety Configuration Time Stamp (SCTS)** – This is a safety related signature that identifies the revision of the device configuration contained in either a Safety Open or in a safety device.

FRS161 The SCTS shall be a Time/Date stamp marking the Time & Date of the configuration creation or change.

FRS160 The SCTS parameter in the Safety Open shall be included in the calculation of the CPCRC.

FRS162 The configuration software shall follow the CIP defined Date and Time format (IEC 1131-3) for setting the signature.

**Safety Function** – The SCTS is used to ensure the identity of the safety application data.

**Safety Configuration Identifier (SCID)** – The SCID is a combination of the SCCRC and the SCTS. The SCTS and SCCRC together serve to uniquely identify a configuration to the user and originator.

FRS163 The SCID = 0 in a Type 2b SafetyOpen shall have special meaning to indicate that the SCID shall not be checked by a target before accepting the connection.

*This special SCID value allows originators who are not concerned about the configuration of a device to make connections without knowing the SCID value.*

FRS103 The safety manual shall contain user instructions that state “If you choose to configure safety connections with an SCID=0, you are responsible for ensuring that originators and targets have the correct configurations”, or similar equivalent wording.



**Initial Time Stamp** – When the Extended safety message format is being used, the Extended format safety segment and Extended SafetyOpen Response is used during connection establishment. Depending on the type of connection (refer to Section 2-2.4), the Initial Time Stamp is either sent in SafetyOpen or in the SafetyOpen Response. The Initial Time Stamp value is used to set up an initial time stamp parameter that is used to detect Time Stamp rollovers.

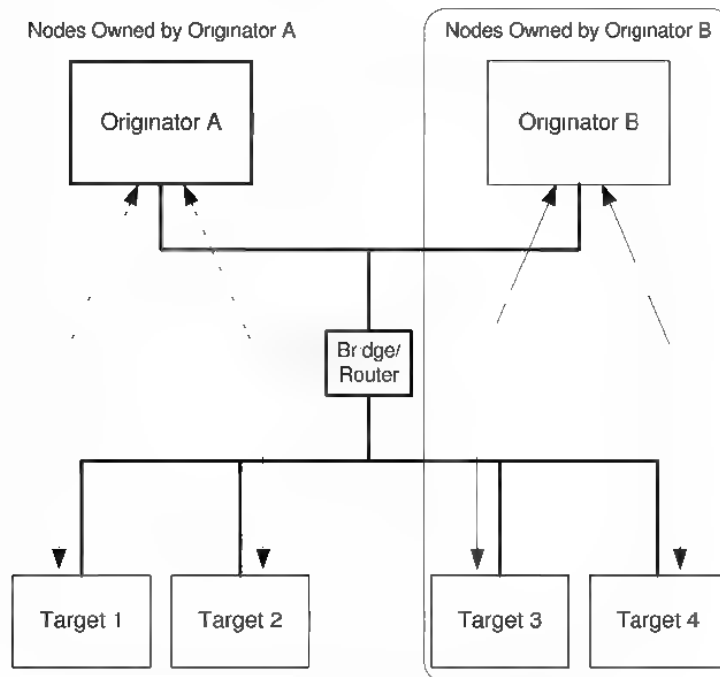
**Initial Rollover Value** – When the Extended safety message format is being used, the Extended format safety segment and Extended SafetyOpen Response is used during connection establishment. Depending on the type of connection (refer to Section 2-2.4), the Initial Time Stamp is either sent in SafetyOpen or in the SafetyOpen Response. The Initial Rollover Value is used to set up the initial Rollover count parameter at the consumer. This value is included in the CRC calculations for the data packets. If the producer and consumer don't use the same value, the consumer CRC checks will fail. This feature provides additional detection capability for inserted messages and out-of-sequence packets.

### 2-3.1.2 Originator-Target Relationship Validation

In the CIP framework, nodes are either originators (nodes that request connections) or targets (nodes that receive connection requests). Controllers and monitors are typically the connection originators. I/O modules (as well as other controllers or monitors) are typically the targets. Targets may support more than one application connection point.

Consider the system depicted in Figure 2-3.2. For the purposes of illustration, assume that all targets are the exact same.

**Figure 2-3.2 Target Ownership**



(Dotted lines represent logical relationships/connections; solid lines are physical connections)



Use of standard bridges/routers and standard lower-layer protocol stacks within the end-nodes offers the opportunity for additional errors that require protection measures that are more stringent than those applied in standard CIP connection establishment. For instance, the bridge/router could mis-direct (e.g. corrupt the destination MACID of) a connection request from originator A intended for Target 1 to Target 2, 3, or 4 instead. Or the source of the error could be the user mis-configuring originator B to connect to Target 1 while originator A is off-line and Target 1 is powered. Without a second level of comparison above the identical electronic key, the connection from originator B to Target 1 would be allowed; at which point originator B's application could place Target 1 (assuming it's an output device) into an incorrect operational state with respect to originator A's commissioned application.

#### 2-3.1.2.1 Detection of Mis-routed Connection Requests

All safety open requests will contain the UNID of the target device. Upon the receipt of a safety open the target must compare the SafetyOpen target UNID (TUNID) with its own UNID.

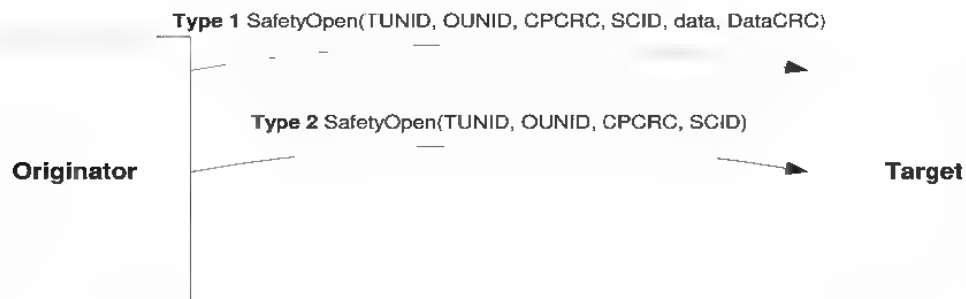
FRS164 If the TUNID to UNID comparison is successful then the SafetyOpen destination is deemed to be correct and processing shall continue.

#### 2-3.1.2.2 SafetyOpen Processing Types

There are two ways of processing the SafetyOpen:

- Destination, Ownership, Configuration data processing and SCID checking
- Destination, Ownership and SCID Checking (Optional)

Figure 2-3.3 SafetyOpen Types



#### 2-3.1.2.3 Ownership Management

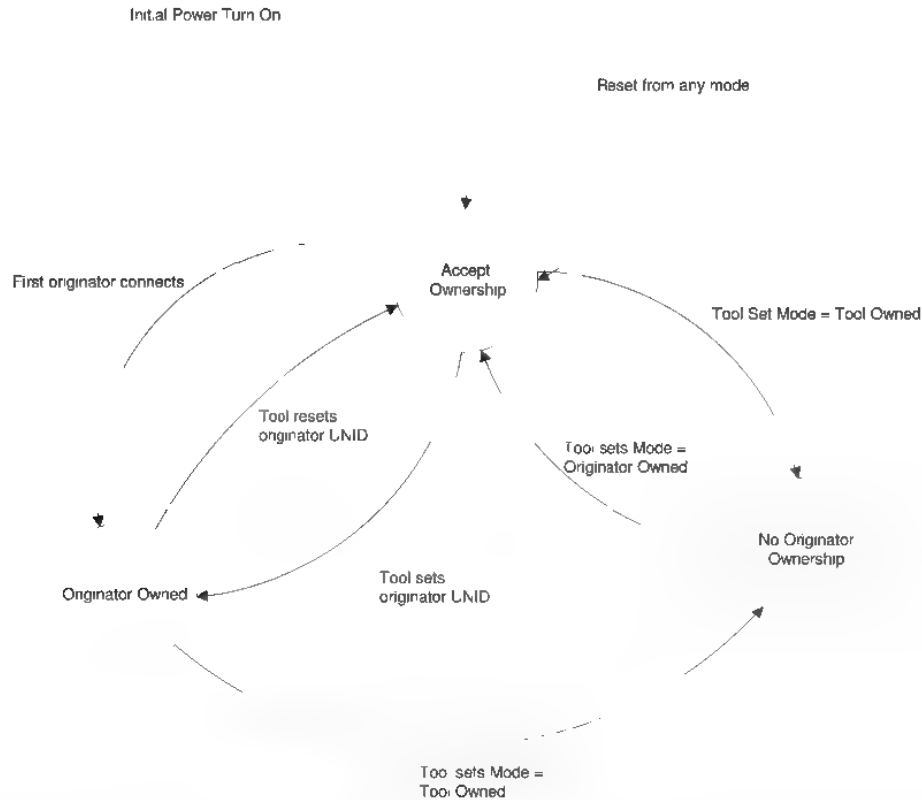
There are two methods of establishing configuration and/or connection ownership, through the configuration tools or from the SafetyOpen. The tool can set the mode to be tool configuration only or originator configuration.

FRS165 Any attempts to reconfigure a device via the SafetyOpen shall be rejected if the tool only configuration mode is selected.



If the originator configured mode is used, the OUNID of the selected originator can be set by the software tool during target device commissioning for added security. Figure 2-3.4 shows the ownership states the methods of transitioning from ownership state to another ownership state.

**Figure 2-3.4 Connection Ownership State Chart**

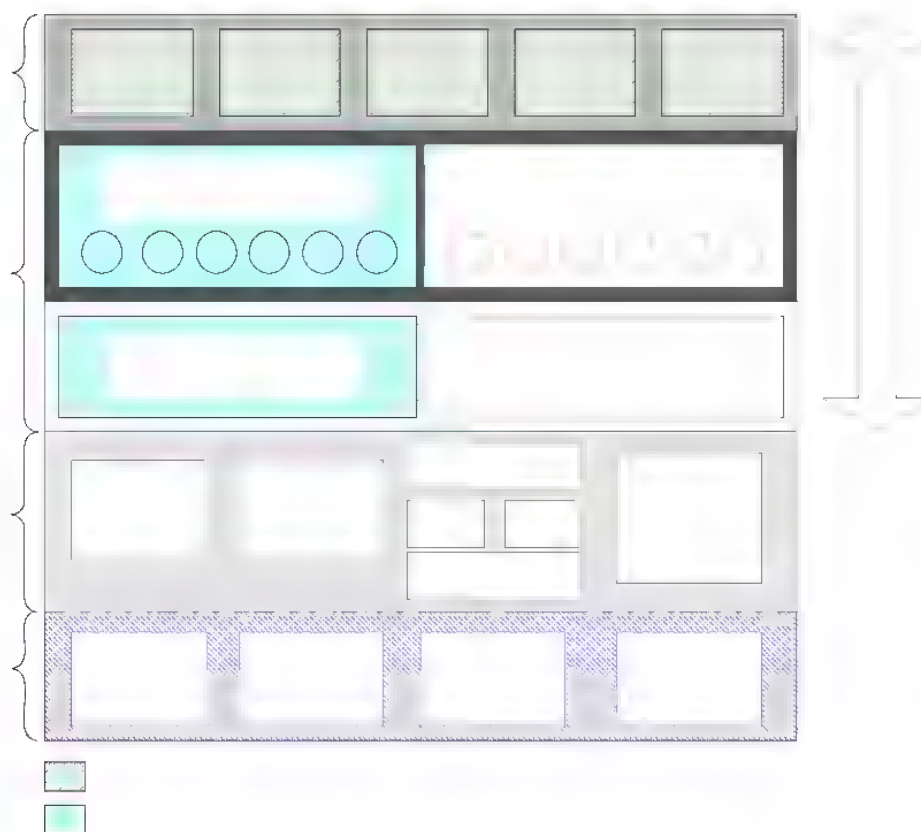


FRS166 To establish ownership of the configuration (and the connection in outputs), in targets the OUNID of the originating device shall be passed to the target in the SafetyOpen service.

FRS167 The OUNID of the originating device shall be stored in non-volatile storage with the configuration to which it is associated.

FRS168 Output devices shall store the OUNID associated with safety outputs to prevent other originators from hijacking outputs inadvertently. Keeping the originator OUNID associated with the configuration will allow multiple devices to connect to an input device and still maintain configuration ownership. Figure 2-3.5 shows the mapping of the safety open UNIDS to the target device originator ownership locations.



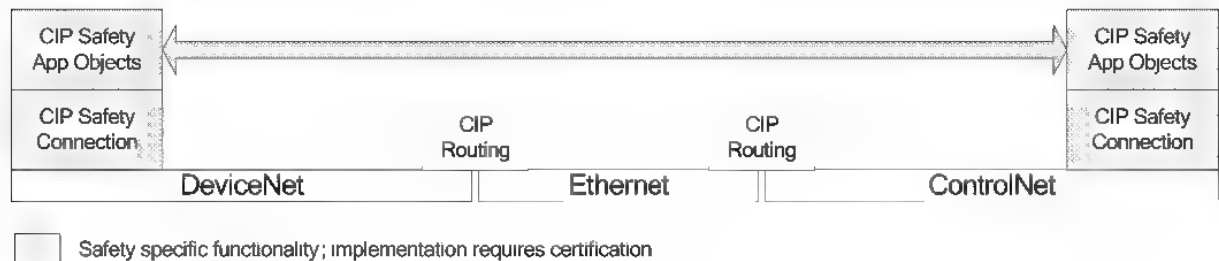




The CIP connection objects are the entities that reconcile the inherent differences between the capabilities and services provided by the underlying networks, exchanging SIL 3 data with the application objects above. For instance, DeviceNet physical layer (i.e. CAN) data frames are limited to a maximum of 8 bytes each. The CIP connection object in a DeviceNet node transports I/O packets larger than 8 bytes via DeviceNet's I/O fragmentation protocol; where as on the higher-capacity CIP networks like EtherNet/IP, the maximum CIP connection size of 511 bytes fits into one physical layer data frame. Once a safety connection is established, the CIP safety application layer exchanges safety data with the safety connection objects<sup>3</sup>.

The safety protocol has been designed to enable end-to-end transport of the safety data without requiring specific knowledge of the safety protocol within or SIL 3 certification of intermediate devices. Figure 2-3.7 depicts an end-to-end connection between a safety node on DeviceNet and a safety node on ControlNet routed across an EtherNet/IP network. The safety connection endpoints enable the application level peers to share SIL 3 data.

**Figure 2-3.7 End-to-End Routing Example**



Safety connection establishment is dynamic on every network. EtherNet/IP originators employ the Forward\_Open service of the connection manager object. The connection path of a Forward\_Open service's data includes a safety network segment for the end node. This safety network segment informs the recipient that a connection object capable of supporting a safety connection must be available in order to accept the request. It also includes data used to specify Connection Object attributes related to its safety behavior (e.g. expectation timers).

FRS169 DeviceNet originators shall use the connection manager Forward\_Open service (with "Safety" Network Segment extension) when the target is not on the local network segment. This case is also referred to as an off-link connection. However, FRS170 DeviceNet originators shall use the Connection Object's Safety Open request for local targets. This differentiation is required since DeviceNet Devices do not support the connection manager object. DeviceNet routers/bridges do, however, support the connection manager object.

FRS172 On DeviceNet, the originator shall establish an explicit messaging connection with the router or end-node to deliver the Forward\_Open or SafetyOpen service requests. The other CIP networks (typically) deliver the Forward\_Open to the UCMM without a connection.

For detailed examples of establishing connections across different CIP networks, refer to Chapter 10 – Bridging and Routing.

<sup>3</sup> Note that standard application objects may consume safety data for standard application use.



#### **2-3.1.3.1 Basic Facts for Connection Establishment**

The basic facts for safety connection establishment are listed below:

- SafetyOpen is a service defined in the Connection Object
  - An explicit messaging connection is required with a path that targets the Connection object
- ForwardOpen is a service defined in the Connection Manager object
  - On CIP networks, this is an unconnected message with a path that targets the Connection Manager object
  - When originating from a DeviceNet originator, an explicit messaging connection is required with a path that targets the Connection Manager object
- One additional set of connection parameters are required in the SafetyOpen to define the 3<sup>rd</sup> connection point in DeviceNet nodes when the connection type is Multi-cast. These parameters are contained in the Safety Segment.
- A null application path is defined to mean either that a connection has no application data, or the SafetyOpen contains no configuration data. Each safety module vendor shall define a null path that is appropriate for their device. A null path shall always use a full path reference (i.e. 20 04 24 [vendor specific instance]). For EDS-based devices, the selected null instance shall be disclosed with an entry in the connection manager section of the EDS file (refer to Chapter 7).
- The application path for the Time Coordination connection shall always be a null path.
- Type 2 SafetyOpens (no configuration data) shall include a null application path for the configuration path.
- There is no path required for the Time Correction connection
- The extra parameters of the 3<sup>rd</sup> connection point are named and have values as follows:
  - Time Correction RPI
  - Time Correction Network Connection ID
  - Connection Type = Multi-cast
  - Priority = High Priority
  - Fixed/Variable = Fixed
  - Connection Size = 6 (fixed Time Correction size)
- The Time Correction Network Connection ID is not part of the CPCRC.
- The standard O→T and T←O connection sizes shall always reflect the 2-connection size for each. In other words, the connection that has the Data and Time Correction packets concatenated will have a size that reflects this combination. It will be the responsibility of bridges and the DeviceNet nodes to calculate the appropriate sizes for the connection resources on DeviceNet (this is made easier in that the size of the 3<sup>rd</sup> connection point, provided in the SafetyOpen message, can be added or subtracted).

## **2-4 Safety Message Data Definitions**

### **2-4.1 Mode Byte**

This section defines the data names and ranges for the Mode Byte



#### **2-4.1.1 Mode\_Byte.Run\_Idle**

FRS179 The Run\_Idle bit value of 0 shall indicate Idle, and the Run\_Idle bit value of 1 shall indicate Run .

FRS180 For single-cast connections, the Run\_Idle bit shall be set to Idle if the producing application is Idle or if the producer is waiting for Time Coordination Message information to be received at the initiation of data production.

FRS181 If Time Coordination Message information has been received; the Run\_Idle bit shall be set to the Run\_Idle status indicated by the producing application.

FRS182 For multi-cast connections, the Run\_Idle bit shall be set to the Run\_Idle status indicated by the producing application.

#### **2-4.1.2 Mode\_Byte.N\_Run\_Idle**

FRS183 The N\_Run\_Idle complement bit shall always be the complement of the Run\_Idle bit .

#### **2-4.1.3 Mode\_Byte.TBD\_2\_Bit**

The Mode\_Byte Reserved TBD\_2\_bit shall be set to 0 by the producer

#### **2-4.1.4 Mode\_Byte.TBD\_2\_Copy**

The TBD\_2\_Copy bit shall always equal to the TBD\_2 bit.

#### **2-4.1.5 Mode\_Byte.Ping\_Count**

The Ping\_Count is used by the consumer to know when it should send a Time Coordination Message.

#### **2-4.1.6 Mode\_Byte.TBD\_Bit**

The Mode\_Byte Reserved TBD\_bit shall be set to 0 by the producer.

FRS190 The Reserved bits shall be ignored by the consumer except for CRC checking and Actual/Complement checking of the TBD\_Bit/N\_TBD\_Bit pair.

#### **2-4.1.7 Mode\_Byte.N\_TBD\_Bit**

FRS191 The N\_TBD\_Bit complement bit shall always be the complement of the TBD\_Bit .

### **2-4.2 Time Stamp Section**

#### **2-4.2.1 Time\_Stamp\_Section.Data\_Time\_Stamp**

The Data\_Time\_Stamp is the time that the Data was sampled for this production.



For single-cast, the Data\_Time\_Stamp is relative to the timer that is on the consumer. For multi-cast, the Data\_Time\_Stamp is relative to the timer that is on the producer.

The Data\_Time\_Stamp is always in 128 uSec increments.

### **2-4.3 Time Coordination Message**

#### **2-4.3.1 Ack\_Byte.Ping\_Response Bit**

The Ping\_Response bit shall indicate that a ping response is being sent on this data production from the safety data consumer on this node.

FRS195 If the Ping\_Response bit is 1, it shall indicate that the Consumer\_Time\_Value, and Ping\_Count\_Reply, are valid on this production .

FRS196 If the Ping\_Response bit is 0, it shall indicate that the Consumer\_Time\_Value, and Ping\_Count\_Reply, are not valid on this production and should be ignored.

#### **2-4.3.2 Ack\_Byte.Consumer\_Time\_Value**

The Consumer\_Time\_Value shall be the safety data consumer's clock value at the time that the message is sent.

The Consumer Time Value is used to derive the Data\_Time\_Stamp for the other directions data production (see FRS18, FRS61, FRS72).

#### **2-4.3.3 Ack\_Byte.Ping\_Count\_Reply**

The Ping\_Count\_Reply is used by each consumer to indicate the ping request that is being responded to (See FRS55).

The Ping\_Count Reply is equal to the value of the Ping\_Count that is being responded to (see FRS55).

#### **2-4.3.4 Ack\_Byte.Time Coordination Reserved bits**

FRS203 The Time Coordination Reserved bits shall be set to 0 by the consumer.

FRS204 The Reserved bits shall be ignored by the producer except for CRC and Ack\_Byte\_2 checking.

### **2-4.4 Time Correction Message**

#### **2-4.4.1 Mcast\_Byte.Consumer\_#**

FRS205 For multi-cast produced data the Multi\_Cast\_Active\_Idle, and Consumer\_Time\_Correction\_Value shall be directed to the consumer indicated by the Consumer\_# field of the message.



The Consumer\_# points to a specific consumer (1 through 15).

FRS206 The Multi\_Cast\_Active\_Idle, and the Consumer\_Time\_Correction\_Value shall be applied by the consumer indicated by the Consumer\_#.

A Consumer\_# of 0 should be used to indicate Multi\_Cast\_Active\_Idle, and the Consumer\_Time\_Correction\_Value are not being transferred if needed.

#### **2-4.4.2 Time\_Correction\_Section.Consumer\_Time\_Correction\_Value**

The Consumer\_Time\_Correction\_Value is used by the multi-cast consumer to correct the Producer\_Time\_Stamp so the resultant time stamp is relative to the consumers' clock (see FRS61).

#### **2-4.4.3 Mcast\_Byte.Multi\_Cast\_Active\_Idle**

FRS209 The Multi\_Cast\_Active\_Idle bit value of 0 shall indicate Idle, and the Multi\_Cast\_Active\_Idle bit value of 1 shall indicate Active.

FRS210 Once the Multi\_Cast\_Active\_Idle bit has been set to Active after 1st data production, this bit shall not be set back to idle until the safety connection is re-initialized .

FRS211 If the consumer sees the Multi-Cast\_Active\_Idle bit transition from Active to Idle, it shall close the connection if the base format is used. If the Extended Format is used and the Max\_Fault\_Count has not been reached, the packet will be dropped, otherwise the connection will be closed.

For multi-cast connections, the Run\_Idle bit of the Mode\_Byte shall be set to the Run\_Idle status indicated by the producing application (see FRS182).

The consumer will need to look at both the Run\_Idle bit and the Multi\_Cast\_Active\_Idle bit.

#### **2-4.4.4 Mcast\_Byte.Time Correction Reserved Bits**

FRS215 The Time Correction Reserved bits shall be set to 0 by the producer .

FRS216 The Reserved bits shall be ignored by the consumer except for CRC checking and MC\_Byte\_2 checking .

### **2-4.5 Safety Data Production**

This section describes the variables used in the production of safety data.

#### **2-4.5.1 Producing Connection Status**

FRS217 The producer connection status shall be determined by the combination of the S\_Connection\_Fault and the Consumer\_Active\_Idle flags (Refer to Table 2-4.1) ,



**Table 2-4.1 Producer Connection Status Determination**

S Connection_Fault	Consumer Active Idle	Producer Connection Status	Comment
Fault	Don't Care	Status= Faulted.	Restart will be required
OK	Idle	Status=Idle	The safety connection has not been activated
OK	Active	Status=Active	The safety connection is active

#### **2-4.5.1.1 Consumer\_Open**

FRS322 For Safety Data producers, Consumer\_Open from the Validator Connection Establishment Engine to the Run Time Validator shall be provided to indicate whether or not a particular consumer of the produced data has an active open connection.

FRS323 If the producer is multi-cast, Consumer\_Open shall exist for each multi-cast consumer, 1 through Max\_Consumer\_Number (indexed from 0 to Max\_Consumer\_Number – 1).

The Consumer\_Open values are OPEN or CLOSED.

FRS324 The Consumer\_Open indication shall act as an enable to the Run Time Validator on a connection by connection basis. After successful connection establishment, Consumer\_Open flags shall transition from Closed to Open.

FRS326 The Validator Connection Establishment engine shall provide an array of consumer connection status resources for multi-cast producers that shall be allocated when a consumer makes a connection.

FRS327 The consumer numbers allocated to a Run-time validator shall be the number sent to the consumer in the connection establishment reply. (refer to 2-4.4.1)

The consumer connection numbers can be treated as an allocation pool that can be re-used and re-allocated as needed.

FRS325 The Validator Connection Establishment Engine shall ensure that the Consumer\_Open resource for any allocated consumer number is set to Closed for a minimum time of (Timeout\_Multiplier.PI +2) \* Ping Interval prior to re-allocating that Consumer number to another connection. This is to ensure that any previously established connections have timed out prior to allocating the number to a new connection.

In other words, if the Validator Connection Establishment Engine sees S\_Connection\_Fault set, it should set Consumer\_Open to CLOSE for a minimum of (Timeout\_Multiplier.PI +2) \* Ping Interval or optionally until that same consumer tries to re-establish the same connection.

#### **2-4.5.1.2 Application\_Run\_Idle**

FRS218 If the producing application is actively controlling data the Run\_Idle indication shall indicate Run. If the producing application is not actively controlling the data and the data should be put in the safety state, the Run\_Idle indication shall indicate Idle.



#### **2-4.5.1.3      Consumer\_Active\_Idle [per consumer]**

Consumer\_Active\_Idle is a run time flag to indicating if valid Time Coordination Message information has been received for this consumer.

FRS219 If a safety data producer is producing data and valid Time Coordination information has been received without errors the Consumer\_Active\_Idle flag shall be equal to Active, otherwise it will be Idle.

FRS220 For multi-cast producers, a Consumer\_Active\_Idle flag shall exist for each multi-cast consumer, numbered 1 through Max\_Consumer\_Number.

The Consumer\_Active\_Idle values are Idle and Active.

FRS221 At connection establishment, all Consumer\_Active\_Idle flags shall be initialized to Idle.

#### **2-4.5.1.4      S\_Connection\_Fault [per consumer]**

S\_Connection\_Fault is a run time flag that indicates whether the safety connection to the consumer is OK or Faulted.

FRS222 An S\_Connection\_Fault indication of Fault, shall indicate that the producer has detected a problem with the safety connection to this consumer or the consumer has indicated an error back to the producer.

FRS223 If the safety data producer has received invalid Time Coordination information, or a Time Coordination information timeout, the S\_Connection\_Fault flag shall be equal to Fault, otherwise it will be OK.

FRS224 For multi-cast producers, a S\_Connection\_Fault flag shall exists for each multi-cast consumer, numbered 1 through Max\_Consumer\_Number.

The S Connection Fault values are OK and Faulted.

FRS225 At connection establishment, all S\_Connection\_Fault variables shall be initialized to OK .

### **2-4.5.2      Producer Input Static Variables**

Producer input static variables are the configuration time parameters that define the operation of the safety protocol producer.

#### **2-4.5.2.1      EPI**

EPI is the expected packet interval. Safety data shall be produced at a rate not exceeding the EPI. The EPI is expressed as an integer value in uSec. The legal range of values is from 100 to 1,000,000 uSec.

For all safety connections, the producer and all consumers shall all have the same EPI.



#### **2-4.5.2.2      Timeout\_Multiplier [per consumer]**

The Timeout\_Multiplier indicates the number of data production retries to use in the connection failure detection equations.

The Base Format limited the Timeout\_Multiplier parameter values from 1 to 4. For the Base Format, the Timeout\_Multiplier value was used for two purposes. They are:

- 1) To calculate the Network Time Expectation.
- 2) To determine the number of ping intervals to wait without Time\_Coordination or Time Correction before declaring a connection fault.

The ExtendedFormat allows a larger Timeout\_Multiplier parameter value to be used to calculate the Network Time Expectation, but the value used to determine the number of ping intervals to wait without Time\_Coordination or Time Correction before declaring a connection fault is still limited to a maximum value of 4. The ExtendedFormat allows the Timeout\_Multiplier parameter value to be any value from 1 to 255.

Throughout this document “Timeout\_Multiplier” will refer to the Timeout\_Multiplier parameter value. “Timeout\_Multiplier.PI” will refer to the value that is derived from “Timeout\_Multiplier” to determine the number of Ping Intervals to wait without Time\_Coordination or Time Correction before declaring a connection fault.

FRS374 For the Base Format, Timeout\_Multiplier.PI shall be equal to the Timeout\_Multiplier. For the ExtendedFormat, Timeout\_Multiplier.PI shall be equal to the smaller of Timeout\_Multiplier or 4, whichever is lower.

FRS229 The Timeout\_Multiplier shall be used to determine how many ping intervals  $(\text{Timeout\_Multiplier.PI} + 2)$  a producer will wait before faulting a consumer who has failed to send a Time Coordination response to a Ping request .

FRS230 The legal range of the base format Timeout\_Multiplier values shall be 1, 2, 3, or 4. The legal range of the Extended Format Timeout\_Multiplier value shall be 1 to 255 so long as the Network Time Expectation value does not exceed 5.8 seconds.

For any producer, each consumer may have a different Timeout\_Multiplier.

#### **2-4.5.2.3      Connection\_Type**

The Connection\_Type specifies whether the connection is single-cast or multi-cast.

FRS231 For any producer, each producer connection and the corresponding consumer connection(s) shall be of the same Connection Type.

#### **2-4.5.2.4      Ping\_Interval\_EPI\_Multiplier**

Ping\_Interval\_EPI\_Multiplier is the number that defines the Ping\_Count\_Interval for a particular connection.



FRS232 The Ping\_Count\_Interval shall be equal to the EPI times the Ping\_Interval\_EPI\_Multiplier.

FRS233 The Ping\_Count\_Interval shall define the rate at which Time Coordination Messages will be requested and sent.

FRS234 The Ping\_Count\_Interval shall define the rate at which Time Correction sections are sent to each consumer.

FRS235 A legal Ping\_Interval\_EPI\_Multiplier shall have a minimum value determined from the equation  $\text{Max\_Consumer Number} * \text{Timeout\_Multiplier.PI} + 15$ , and a maximum less than 1000. The default values shall be 100 for Multi-cast and 19 for Single Cast

The Ping\_Interval\_EPI\_Multiplier should be set large enough to allow the ping request to be recognized by the server(s), to send the Time Coordination section(s), and all Time\_Coordination section(s) to arrive and be processed by the client. All these steps must occur before the Ping\_Interval time expires.

FRS237 All Time\_Correction sections shall be sent out by the client within the Ping\_Interval.

FRS318 The Ping\_Interval\_EPI\_Multiplier shall be set small enough to ensure that the Ping\_Count\_Interval is less than or equal to 100 seconds.

A ping count interval of greater than 100 seconds may contribute greater than .5 seconds to the clock drift, depending upon the Timeout\_Multiplier.PI for that connection. To prevent clock size wrap around, .5 seconds has been set at the maximum value that can be allocated to clock drift.

The following variables will be used to calculate the Ping\_Interval\_EPI\_Multiplier.

- **C\_TO\_S\_Delay\_Multiplier** equals the time as a multiplier of the EPI time that is equal to the time for message initiation, transport delay, and reception processing delay of the Data Message or the Time\_Correction Message from the client to the server.
- **S\_TO\_C\_Delay\_Multiplier** equals the time as a multiplier of the EPI time that is equal to the time for message initiation, transport delay, and reception processing delay of the Time\_Coordination Message from the server to the client.
- **Ping Rcved TO Time Coord Sent Multiplier** equals the time as a multiplier of the EPI time that is allowed from the time that the EPI reception that should trigger the sending of the Time Coordination message until the message should be sent.
- **Time Coord Rcved TO Time Coord Processed Multiplier** equals the time as a multiplier of the EPI time that is allowed from the time that the Time Coordination message is received until the Time Coordination message is processed.

For a single network it is reasonable to expect that C\_TO\_S\_Delay\_Multiplier and S\_TO\_C\_Delay\_Multiplier are less than 2 times the EPI.

For a bridged network depending upon the number of hops, it is reasonable to expect that C\_TO\_S\_Delay\_Multiplier and S\_TO\_C\_Delay\_Multiplier are less than 4 times the EPI. 4 will be used as the minimum value for C\_TO\_S\_Delay\_Multiplier and S\_TO\_C\_Delay\_Multiplier when calculating the minimum value for Ping\_Interval\_EPI\_Multiplier.



The C\_TO\_S\_Delay\_Multiplier and S\_TO\_C\_Delay\_Multiplier EPI multipliers should not have to consider messages being lost on the network. The Timeout\_Multiplier will account for lost messages.

Ping\_Rcvd\_TO\_Time\_Coord\_Sent\_Multiplier should be less than 3 times the EPI.

Time\_Coord\_Rcvd\_TO\_Time\_Coord\_Processed\_Multiplier should be less than 3 times the EPI.

The equation to determine the minimum Ping\_Interval\_EPI\_Multiplier is:

$$\begin{aligned} & [ \{ \text{the maximum Timeout\_Multiplier.PI (C\#) for all consumers} \} * \text{Max\_Consumer\_Num} ] + \\ & \text{C\_TO\_S\_Delay\_Multiplier} + \text{S\_TO\_C\_Delay\_Multiplier} + \\ & \text{Ping\_Rcvd\_TO\_Time\_Coord\_Sent\_Multiplier} + \\ & \text{Time\_Coord\_Rcvd\_TO\_Time\_Coord\_Processed\_Multiplier} + 1 \\ & \text{where C\# = Consumer\_Number} \end{aligned}$$

Using 4 as the constant used for C\_TO\_S\_Delay\_Multiplier and S\_TO\_C\_Delay\_Multiplier and 3 as the constant used for Ping\_Rcvd\_TO\_Time\_Coord\_Sent\_Multiplier and Time\_Coord\_Rcvd\_TO\_Time\_Coord\_Processed\_Multiplier:

FRS239 The Ping\_Interval\_EPI\_Multiplier shall be greater than or equal to:

$$[ \{ \text{the Max Timeout\_Multiplier.PI (C\#) for all consumers} \} * \text{Max\_Consumer\_Num} ] + 15$$

This relationship will be relied upon by the producer when sending Time Correction messages. For C\_TO\_S\_Delay\_Multiplier and S\_TO\_C\_Delay\_Multiplier values greater than 4 , and Ping\_Rcvd\_TO\_Time\_Coord\_Sent\_Multiplier and Time\_Coord\_Rcvd\_TO\_Time\_Coord\_Processed\_Multiplier greater than 3, the Ping\_Interval\_EPI\_Multiplier must be increased accordingly.

The Ping\_Interval\_EPI\_Multiplier must be less than: 100 Sec / EPI.

The Ping\_Interval\_EPI\_Multiplier range of 75 to 100 will meet the restrictions defined above for all legal values defined below:

- a) EPI, .0001 to 1 Sec
- b) Timeout\_Multiplier.PI, 1 to 4
- c) Max\_Consumer\_Num, 1 to 15
- d) C\_TO\_S\_Delay\_Multiplier and S\_TO\_C\_Delay\_Multiplier, 1 to 4.
- e) Ping\_Rcvd\_TO\_Time\_Coord\_Sent\_Multiplier and  
Time\_Coord\_Rcvd\_TO\_Time\_Coord\_Processed\_Multiplier, 1 to 3

For any producer, the producer and all consumers must have the same Ping\_Interval\_EPI\_Multiplier.



#### **2-4.5.2.5      Max\_Consumer\_Number**

Max\_Consumer\_Number is the maximum number of consumers that may be configured.

FRS241 If the connection type is single-cast, the Max\_Consumer\_Number shall be equal to 1.

FRS242 For multi-cast connections, the legal range of Max\_Consumer\_Number values shall be from 1 through 15.

For any producer, the producer and all consumers must have the same Max\_Consumer\_Number.

#### **2-4.5.2.6      Time\_Coord\_Msg\_Min\_Multiplier [per consumer]**

Time\_Coord\_Msg\_Min\_Multiplier is the minimum number of 128 uSec increments it could take for a Time Coordination Message to traverse from the consumer to the producer. A value other than 0 for this variable will allow the time stamp to be more accurate.

FRS243 The default for the Time\_Coord\_Msg\_Min\_Multiplier shall be 0, and the legal range of values shall be from 0 through 7813, which equates to 0 through 1 Sec .

This is the minimum number of 128 uSec increments it could take:

##### **From:**

The consumer capturing Consumer\_Time\_Value from the Consumer\_Clk\_Count for the Time Coordination Message,

##### **To:**

The producer capturing the Producer\_Rcvd\_Time\_Value upon reception of the Time Coordination Message

For any producer, each consumer may have a different Time\_Coord\_Msg\_Min\_Multiplier.

#### **2-4.5.3      Producer Connection Derived Static Variables**

These static variables are derived from the input static variables. These static variables are derived at configuration time and are used during producer processing.

##### **2-4.5.3.1      Time\_Drift\_Per\_Ping\_Interval [per consumer]**

The Time\_Drift\_Per\_Ping\_Interval is a value in 128 uSec increments that represents the worst-case time drift due to crystal inaccuracy during one Ping Interval.

FRS244 The clocks in safety devices shall be accurate to within 0.02% (0.0002) . Considering positive and negative drift the clocks will remain within 0.0002 \*2 of each other or 0.0004 (.04%).

The legal range of values is from 1 through 313, since the Ping Interval is limited to 100 seconds (100,000,000 uSec).



The Time\_Drift\_Per\_Ping\_Interval calculation is done with time references in uSeconds (EPI is the EPI in uSecond units). The Time\_Drift\_Per\_Ping\_Interval calculation is done in 32 bit integer math with the result being a 16 bit integer. The divide by 320000 factor in the Time\_Drift\_Per\_Ping\_Interval calculation is equivalent to multiplying by the worst case clock drift of .0004 and dividing by 128 uSeconds.

Time\_Drift\_Per\_Ping\_Interval (C#)

= Roundup [EPI \* Ping\_Interval\_EPI\_Multiplier / 320000]

**OR** 1 whichever is greater.

where C# = Consumer\_Number

**Example:** if EPI = 8000us and Ping\_Interval\_EPI\_Multiplier = 100

Time\_Drift\_Per\_Ping\_Interval(C#) = 3 (384 us)

#### **2-4.5.3.2 Connection\_Correction\_Constant [per consumer]**

Connection\_Correction\_Constant is a value in 128 uSec increments that is subtracted from the time stamp to represent the worst case error due to:

1. Time drift
2. The asynchronous nature of the producer and consumer clocks
3. The minimum time for the Time Coordination Message to traverse from the consumer to the producer

Connection\_Correction\_Constant [C#] =

Time\_Drift\_Constant + 1 - Time\_Coord\_Msg\_Min\_Multiplier [C#]

where C# = Consumer\_Number

Note that the Time\_Drift\_Constant (which is part of the Connection\_Correction\_Constant calculation) is a value in 128 uSec increments that is subtracted from the time stamp to represent the worst-case time drift due to crystal inaccuracy. The producer and consumer clocks must be accurate to within 0.02% (0.0002). Considering positive and negative drift the clocks will remain within 0.0002 \*2 of each other or 0.0004 (.04%).

The legal range of values for the Time\_Drift\_Constant is from 1 through 1563.

The Time\_Drift\_Constant calculation is done with time references in uSeconds (EPI is the EPI in uSecond units).

FRS245 The Time\_Drift\_Constant calculation shall be done using at least 32 bit integer math with the result being a 16 bit integer.

The 1/320000 factor in the Time\_Drift\_Constant calculation is equivalent to multiplying by the worst case clock drift of .0004 and dividing by 128.

Time\_Drift\_Constant (in 128 uSec counts)



= Roundup [(Timeout\_Multiplier.PI (C#)+1) \* EPI\_in\_usec \* Ping\_Interval\_EPI\_Multiplier / 320000]

**OR** 1 whichever is greater.

**Example:** Timeout\_Multiplier.PI =2, EPI(in uSec)=8000, Ping\_Interval\_EPI\_Multiplier=100

Time\_Drift\_Constant = 8, Time\_Coord\_Msg\_Min\_Multiplier = 2

Connection\_Correction\_Constant = 7

### **2-4.5.3.3 Time\_Coord\_Response\_EPI\_Limit [per consumer]**

The Time\_Coord\_Response\_EPI\_Limit is used to ensure that a Time Coordination message that returns to the producer requesting it, arrives within a time sufficient to prevent the possibility of a 16 bit time stamp math rollover. There are other checks to insure that the Time\_Coordination message returns within the Ping Interval and that not too many Time Coordination messages are dropped.

Time\_Coord\_Response\_EPI\_Limit (a value in EPI increments) is used to determine if the time between the incrementing of the Ping Count and the Time Coordination response is less than a time value that is equal to:

$\{5 + [\text{Time\_Coord\_Msg\_Min\_Multiplier} * .000128] + [\text{EPI} * (\text{Consumer\_Num} - 1)]\}$  in seconds.

To keep the reaction time less than 5.8 seconds, the transport time of the safety message plus the transport time of the Time Coordination message minus the minimum Time Coordination message  $[\text{Time\_Coord\_Msg\_Min\_Multiplier} / .000128]$  must be less than 5.0 seconds if the worst case drift of .5 seconds is considered. This procedure keeps the Connection\_Correction\_Constant within range to prevent rollover at the age check.

The legal range of values for Time\_Coord\_Response EPI Limit is between 5 to 1000.

FRS320 If the Consumer\_Active\_Idle[Consumer\_Num-1] == Active, Producers shall check that Time Coordination Messages are received within the Time\_Coord\_Response\_EPI\_Limit and fault the connection if they are not.

The Time\_Coord\_Response\_EPI\_Limit calculation is done with time references in uSeconds (EPI is the EPI in uSecond units).

FRS247 The Time\_Coord\_Response\_EPI\_Limit calculation shall be done in 32 bit integer math with the result being a 16 bit integer.

Time\_Coord\_Response\_EPI\_Limit [C#] =

$\text{Roundup}(\{5000000 + [\text{Time\_Coord\_Msg\_Min\_Multiplier}[C#] * 128] + [\text{EPI} * (C\# - 1)]\} / \text{EPI})$

**OR** 1000 whichever is lower.

where C# = Consumer\_Number



**Example:** if Time\_Coord\_Msg\_Min\_Multiplier = 2 (256 us), EPI = 8000 usec, and

Consumer\_Number = 1.

Time\_Coord\_Response\_EPI\_Limit = 626

#### **2-4.5.4 Producer Dynamic Variables**

The following dynamic variables are producer variables that are common to all multi-cast consumers consuming the produced data.

##### **2-4.5.4.1 Producer\_Clk\_Count**

FRS248 Each product that produces or consumes safety data shall derive a periodic timer that increments a counter (Producer\_Clk\_Count or Consumer\_Clk\_Count) every 128 uSecs. Producer\_Clk\_Count must have tolerances and stability within +/- 0.02% or 200 PPM.

FRS249 Producer\_Clk\_Count (and Consumer\_Clk\_Count) shall be 16 bits in length.

##### **2-4.5.4.2 Producer\_Safe\_Data\_TS**

FRS250 Producer\_Safe\_Data\_TS is a variable that shall be set equal to Producer\_Clk\_Count at the point in time that the Safety\_Data for a particular EPI data production is sampled and captured from the producing application.

This location that is sampled is the point where the producing applications time monitoring responsibilities end and the safety protocols time monitoring responsibilities begin. This value is used as the time stamp for the data that is relative to the producer's clock. The legal range of values is from 0 through 65535.

##### **2-4.5.4.3 Data\_Time Stamp**

Data\_Time\_Stamp is the time that the safety data was sampled relative to the consumer's clock.

The Data\_Time\_Stamp value is derived by the producer for a single-cast production.

The Data\_Time\_Stamp value is derived by the consumer in multi-cast connections.

The legal range of values is from 0 through 65535.

For single-cast:

Data\_Time\_Stamp = Producer\_Safe\_Data\_TS + Consumer\_Time\_Correction\_Value

For multi-cast:

Data\_Time\_Stamp = Producer\_Safe\_Data\_TS



#### **2-4.5.4.4      Ping\_Interval\_EPI\_Count**

Ping\_Interval\_EPI\_Count is a count of the number of EPI time periods between the incrementing of the Ping\_Count.

Ping\_Interval\_EPI\_Count is used to control Ping Intervals and to control the sending of Time Correction messages for multicast producers.

The Ping\_Interval\_EPI\_Count is incremented at every data production. When Ping\_Interval\_EPI\_Count reaches the Ping\_Interval\_EPI\_Multiplier, it is reset to 0, and the Ping count is incremented.

FRS257 For multi-cast producers, Time Correction messages shall be sent to the n consumers (where n = 1 thru Max\_Consumer\_Number), on the 9<sup>th</sup> thru n+9 productions of the Ping Interval.

FRS258 The Time Correction message of consumer 1 shall be sent first, followed by 2, 3,...,n.

The intent of this logic is to distribute the Time Correction messages at a point within the Ping Interval, when the Time Coordination messages should have already arrived and been processed for this Ping Interval.

The legal range of values is from 0 through Ping\_Interval\_EPI\_Multiplier.

At connection establishment, the Ping\_Interval\_EPI\_Count variable should be set to 0x0000.

#### **2-4.5.4.5      RR\_Con\_Num\_Index\_Pntr**

RR\_Con\_Num\_Index\_Pntr is a variable used by the multi-cast producer to control the checking of the timeout on the reception of Time Coordination messages, and to control the sending of Time Correction messages.

A value greater than or equal to Max\_Consumer\_Number, indicates that the Time Coordination timeouts are not being checked on this EPI, and no Time Corrections are being sent based on this EPI.

A value of 0 thru (Max\_Consumer\_Number -1) indicates which Time Coordination timeouts will be checked (Consumer\_Number = RR\_Con\_Num\_Index\_Pntr + 1) on this EPI and which Time Correction message (Consumer\_Number = RR\_Con\_Num\_Index\_Pntr + 1) may be sent based on this EPI production.

#### **2-4.5.4.6      Time\_Drift\_Since\_Last\_Time\_Coord [per consumer]**

Time\_Drift\_Since\_Last\_Time\_Coord is a variable that contains the maximum time drift that could have occurred between the producer and the consumer since the last time that a Time Coordination message was received. This value is equal to the Ping\_Int\_Since\_Last\_Time\_Coord\_Msg\_Count times Time\_Drift\_Per\_Ping\_Interval. The Ping\_Int\_Since\_Last\_Time\_Coord\_Msg\_Count is the number of Ping Intervals since the last Time Coordination message.



FRS260 The `Time_Drift_Since_Last_Time_Coord` shall be used to indirectly determine if the delay of the Time Coordination message received is significantly greater than the delay for the Time Coordination message previously being used.

The Delay of the Time Coordination message has an impact on the Consumer Time Correction Value that will be used. Greater delays in the Time Coordination message cause the `Time_Stamp` to be compensated in a negative direction, causing the age of the data to be greater.

If the difference between the delays of two consecutive `Time_Coordination` messages is greater than the EPI time, the later `Consumer_Time_Correction_Value` may cause the `Time_Stamp` of the next EPI production to actually be less than the `Time_Stamp` of the previous EPI production. This would result in a sequence error at the consumer of the `Time_Stamp`.

FRS261 Since the maximum time drift is controlled, either the `Last_Time_Correction_Value` minus the `Time_Drift_Since_Last_Time_Coord` value, or the `New_Time_Correction_Value` value, shall be sent to the consumer, whichever is better.

This prevents long delays in the delivery of a Time Correction messages from causing sequence errors in the Time Stamp checks.

FRS262 For multi-cast producers, a `Time_Drift_Since_Last_Time_Coord` shall exist for each multi-cast consumer, 1 through `Max_Consumer_Number`.

The legal range of values is from 0 through  $3910 = (5 * 782)$ .

#### **2-4.5.4.7 Worst\_Case\_Consumer\_Time\_Correction\_Value**

This a temporary value used to determine the `Consumer_Time_Correction_Value`.

#### **2-4.5.5 Producer Per Consumer Dynamic Variables**

The following dynamic variables are producer variables that exist for each consumer. For single-cast connections there will be only one variable for each consumer. For multi-cast there will be `Max_Consumer_Number` of variables, each representing an individual consumer.

##### **2-4.5.5.1 Consumer\_Time\_Value[per consumer]**

`Consumer_Time_Value` is the time value sent to the producer of a time stamp from the consumer or in the Time Coordination Message. A `Consumer_Time_Value` is received for each multi-cast consumer, 1 through `Max_Consumer_Number`.

The legal range of values is from 0 through 65535.

FRS263 At connection establishment, all `Consumer_Time_Value` variables shall be set to 0x0000.

FRS360 When a Time Coordination Message is received; the producer shall check that the `Time_Coordination_Section.Consumer_Time_Value` is not the same as the last received `Consumer_Time_Value`. If it is, the Time Coordination message shall not be processed.



This check will prevent a producer from processing network generated duplicate Time Coordination messages.

#### **2-4.5.5.2      Producer\_Rcvd\_Time\_Value[per consumer]**

FRS264 Producer\_Rcvd\_Time\_Value shall be set equal to Producer\_Clk\_Count at the point in time that the Time Coordination Message information is received from the time stamp of the consumer.

FRS265 A Producer\_Rcvd\_Time\_Value shall exist for each multi-cast consumer, 1 through Max\_Consumer\_Number.

The legal range of values is from 0 through 65535.

FRS266 At connection establishment, all Producer\_Rcvd\_Time\_Value variables shall be set to 0x0000.

#### **2-4.5.5.3      Consumer\_Time\_Correction\_Value[per consumer]**

The Consumer\_Time\_Correction\_Value exists only for multi-cast safety connections.

The Consumer\_Time\_Correction\_Value is added to the producer's data time stamp to allow the result to indicate the worst-case earliest time that the data could have been sampled relative to the consumer's clock value.

A Consumer\_Time\_Correction\_Value exists for each multi-cast consumer, 1 through Max\_Consumer\_Number.

**Consumer Time Correction Value [Consumer Number]**

**= Consumer\_Time\_Value [Consumer Number]  
- Producer\_Rcvd\_Time\_Value [Consumer Number]  
- Connection Correction Constant [Consumer Number]**

The legal range of values is from 0 through 65535.

FRS268 At connection establishment, all Consumer\_Time\_Correction\_Value variables shall be set to 0x0000.

#### **2-4.5.5.4      Ping\_Int\_Since\_Last\_Time\_Coord\_Msg\_Count[per consumer]**

Ping\_Int\_Since\_Last\_Time\_Coord\_Msg\_Count is a run time count of the number of Ping Intervals between Time Coordination Messages for a particular consumer.

FRS269 For single-cast connections, the Ping\_Int\_Since\_Last\_Time\_Coord\_Msg\_Count shall be incremented at the 8<sup>th</sup> EPI production of the Ping Interval.

FRS270 For multi-cast connections, Ping\_Int\_Since\_Last\_Time\_Coord\_Msg\_Count shall be incremented every  $(8 + n)^{\text{th}}$  EPI production of the Ping Interval, where n equals the consumer number - 1.



FRS271 The Ping\_Int\_Since\_Last\_Time\_Coord\_Msg\_Count for any particular consumer shall be reset upon the reception of a valid Time Coordination Message from that consumer .

FRS272 If the Ping\_Int\_Since\_Last\_Time\_Coord\_Msg\_Count ever exceeds the Timeout\_Multiplier.PI + 2 for a connection, that consumer shall be set to the faulted state .

If the producer is multi-cast, Ping\_Int\_Since\_Last\_Time\_Coord\_Msg\_Count exists for each multi-cast consumer, 1 through Max\_Consumer\_Number.

The legal range of values is from 0 through 5.

FRS273 At connection establishment, all Ping\_Int\_Since\_Last\_Time\_Coord\_Msg\_Count variables shall be set to 0x0000 .

#### **2-4.5.5.5 Producer\_Fault\_Counter**

For the Extended Format, this is a temporary value used to track the number of data integrity errors the producer detects on the reception of Time Coordination messages. This value is incremented every time a Time Coordination message is detected with a data integrity error. This value will be reset to 0 every hour. If this value reaches the Max\_Fault\_Number for that consumer (SafetyOpen Parameter), the connection will be closed.

### **2-4.6 Consumer Data Variables**

This section describes the data variables that will be used by a safety consumer to receive safety data. The internal variables are defined in an attempt to define the behavior of the safety data consumer upon the reception of safety data.

#### **2-4.6.1 Consuming Connection Status**

The status of the safety data and the safety connections can be derived from the S\_Con\_Flt\_C\_Out, the S\_Run\_Idle\_Out flags, and the Init\_Complete\_Out. Table 2-4.2 summarizes the logic that the consuming application can use to drive the safety data and status.

**Table 2-4.2 Consuming Safety Connection Status**

<b>S_Con_Flt_C_Out</b>	<b>S_Run_Idle_Out</b>	<b>Init_Complete_Out</b>	<b>Connection Status</b>	<b>Description</b>
Fault	X	X	Faulted	The safety connection is faulted. Restart will be required. Put the data in safety state.
OK	Idle	0	Idle	The safety connection has not been activated. Put the data in safety state.
OK	Idle	1	Idle	The safety connection has been activated and the producing application is in Idle Mode. Put the data in safety state
OK	Active	0	N/A	Illegal condition – can not occur
OK	Active	1	Run	The safety connection is active and the



				producing application is in Run Mode. Use the data
--	--	--	--	---

FRS274 The Connection Status of consuming safety connections shall be indicated as shown in Table 2-4.2 of Volume 5 Chapter 2

#### **2-4.6.1.1 S\_Con\_Flt\_C\_Out**

S\_Con\_Flt\_C\_Out indicates to the consuming application if it should put the safety data in a safety state because the safety validators have detected a safety connection fault. S\_Con\_Flt\_C\_Out\_Temp is used as a temporary location for this variable.

The S\_Con\_Flt\_C\_Out values are OK and Fault.

FRS275 At connection establishment, all S\_Con\_Flt\_C\_Out flags shall be set to OK.

#### **2-4.6.1.2 S\_Run\_Idle\_Out**

FRS276 S\_Run\_Idle\_Out shall be used by a consuming application to determine if it should put the safety data in a safety state.

FRS277 The S\_Run\_Idle\_Out shall indicate Idle when either the producing application is Idle or the Safety Validators are not active. S\_Run\_Idle\_Out\_Temp is used as a temporary location for this variable.

The S\_Run\_Idle\_Out values are IDLE or RUN.

FRS278 At connection establishment, all S\_Run\_Idle\_Out flags shall be set to Idle.

#### **2-4.6.1.3 Init\_Complete\_Out**

The Init\_Complete\_Out flag indicates whether the time stamp initialization is complete or not.

FRS279 S\_Run\_Idle\_Out shall be in Idle until Init\_Complete\_Out is set to a 1.

FRS280 After the connection is first established, the consuming application shall close base format connections if initialization is not completed within 10 seconds. The consuming application shall close Extended connections if initialization is not completed within 8.3 seconds.

The S\_Run\_Idle\_Out will indicate Idle, regardless of this timeout. If the timeout is incorporated, the error must be latched until the safety connection is closed and a re-open attempt is made.

FRS281 For Single-Cast, the flag Init\_Complete\_Out shall indicate that the first time coordination message has been received by the producer and the producer is producing data with the time stamp relative to the consumer's clock.

FRS282 For Multi-Cast, the flag Init\_Complete\_Out shall indicate that the first time coordination message has been received by the producer for this consumer and the consumer has received the 1<sup>st</sup> time correction message based on the time coordination information.



FRS283 For Single-Cast, the Data\_Time\_Stamp value shall be set to 0 by the producer until the first time coordination section is received.

FRS284 If the consumer receives the Data\_Time\_Stamp as a value other than 0, the time stamp checking shall be enabled.

FRS285 For Multi-Cast, the Data\_Time\_Stamp value shall always be set equal to the producers clock.

FRS286 The Multi-cast consumer shall enable time stamp checking upon the reception of the first time correction section.

FRS287 For all consumed safety connections, time stamp checking shall be enabled if the mode of the data indicates Run.

Init\_Complete\_Out\_Temp is used as a temporary location for this variable.

## **2-4.6.2 Consumer Input Static Variables**

### **2-4.6.2.1 Connection\_Type**

Connection\_Type specifies whether the connection is single-cast or multi-cast. For any producer, the producer and all consumers must have the same Connection Type.

### **2-4.6.2.2 Consumer Num**

Consumer \_Num is the number a Multi-cast Consumer is assigned (1 through 15).

### **2-4.6.2.3 Network\_Time\_Expectation\_Multiplier**

Network\_Time\_Expectation\_Multiplier is the maximum number of 128 uSec increments that a consumer should allow the age of the safety data to reach.

FRS288 If consumed safety data becomes older then Network\_Time\_Expectation\_Multiplier, the safety data shall be put in the safety state .

The legal range of values is from 0 through 45313, which equates to 0 through 5.8 Seconds. The 5.8 second limit, the requirement to check the Network\_Time\_Expectation at least once every 2 seconds, and the maximum Time Correction value change (or maximum clock drift) for a Ping interval is .5 Seconds, will ensure that the clock base value of 8.388 seconds (0xFFFF \* .000128) does not wrap around between checks.

The actual network reaction time may be this value in time plus 1 EPI if the producing application is asynchronous to the data production, since in this case the data may be one EPI old at the time that the time stamp was placed on the produced data. See section 2-7 for a complete explanation of the network safety time.



#### **2-4.6.2.4      Timeout\_Multiplier**

Timeout\_Multiplier is the number of data production retries to use for in network fault detection equations. This is the same value used by the producer. See section 2-4.5.2.2 for a further description

#### **2-4.6.2.5      Ping\_Interval\_EPI\_Multiplier**

Ping\_Interval\_EPI\_Multiplier is the number that defines the Ping\_Count\_Interval for a particular connection. The Ping\_Count\_Interval is equal to the EPI times the Ping\_Interval\_EPI\_Multiplier.

This is the same value used by the producer. See section 2-4.5.2.4 for a further description

#### **2-4.6.3      Consumer Connection Derived Static Variables**

These static variables are derived from the input static variables. These static variables are derived at configuration time and are used during producer processing.

#### **2-4.6.4      Consumer Dynamic Variables**

The following dynamic variables are consumer variables that are used to describe safety data reception.

##### **2-4.6.4.1      Consumer\_Clk\_Count**

Each product that consumes safety data will derive a periodic timer that increments a counter (Consumer\_Clk\_Count) every 128 uSecs (see FRS 248).

Consumer\_Clk\_Count will have tolerances and stability within +/- 0.02% or 200 PPM because safety devices are required to have clocks with this tolerance.

Consumer\_Clk\_Count shall be 16 bits in length (see FRS249).

FRS340 If the difference between the Consumer\_Clk\_Count at the time that the data is given to the consuming application for use and the Data Time Stamp sent with the data exceeds the Network\_Time\_Expectation\_Multiplier, the S\_Con\_Flt\_C\_Out flag shall be set to "Faulted" to indicate that the Reaction Time for the network has not been met.

##### **2-4.6.4.2      Last\_Ping\_Count**

FRS291 A consumer of safety data shall be able to detect when the Ping\_Count has changed. This 2 bit variable is used to hold the Ping\_Count for comparison on the next reception.

FRS292 Since the initial production of safety data should have a Ping\_Count of 00, the Last\_Ping\_Count shall be initialized to 11.



#### **2-4.6.4.3 Time\_Coordination\_Count\_Down**

FRS294 The Time\_Coordination\_Count\_Down shall be set equal to the Consumer\_Number upon the reception of a multi-cast ping request.

FRS295 The consumer shall decrement the Time Coordination Count Down value on subsequent reception of consumed data until the value is equal to 0 .

FRS296 If the value is equal to 1 during any reception of consumed data, the ping response for that consumer shall be sent .

The intent of this logic is to distribute the multi-cast ping responses.

FRS297 When a connection is first established, the Time Coordination Count Down shall be initialized to 0.

#### **2-4.6.4.4 Corrected\_Data\_Time\_Stamp**

This value is the consumer copy of the received Data\_Time\_Stamp.

FRS298 For multi-cast, the Corrected Data\_Time\_Stamp shall be a sum of the received time stamp and the Last\_Rcvd\_Time\_Correction\_Value .

#### **2-4.6.4.5 Last\_Data\_Time\_Stamp**

This value is the saved copy of the received Data\_Time\_Stamp.

FRS299 The Last\_Data\_Time\_Stamp shall be used to detect out of sequence messages .

FRS300 When a connection is first established, the Last\_Data\_Time\_Stamp shall be initialized to 0.

#### **2-4.6.4.6 Last\_Rcvd\_Multi\_Cast\_Active\_Idle**

This value is the saved copy of the received Multi\_Cast\_Active\_Idle.

FRS301 The Last\_Rcvd\_Multi\_Cast\_Active\_Idle shall be used to detect a change from active to idle, which would be considered a fault.

FRS302 When a connection is first established, the Last\_Rcvd\_Multi\_Cast\_Active\_Idle shall be initialized to 0.

#### **2-4.6.4.7 Last\_Rcvd\_Time\_Correction\_Value**

This value is the saved copy of the received Time\_Correction\_Value.

FRS303 The Last\_Rcvd\_Time\_Correction\_Value shall be used to correct the Time Stamp for each multi-cast consumer.

FRS304 When a connection is first established, the Last\_Rcvd\_Time\_Correction\_Value shall be initialized to 0.



#### **2-4.6.4.8      Time\_Correction\_Ping\_Interval\_Count**

This value indicates the number of Ping Intervals that have occurred since the last reception of a Time Correction message.

FRS305 If the Time\_Correction\_Ping\_Interval\_Count is greater than the Timeout\_Multiplier.PI + 1, the connection shall be faulted.

#### **2-4.6.4.9      Time\_Correction\_Received\_Flag**

This flag indicates that a Time\_Correction message has been received for this multicast consumer and the time stamp can be corrected and checked.

FRS306 When the Time\_Correction\_Received\_Flag is set, the consumer shall begin the process of correcting and checking the time stamp.

#### **2-4.6.4.10     Data\_Age**

This value indicates the worst case age of the data in 128 uSec increment.

Data Age is determined at the point a Consuming Application samples the data. In Application-Triggered consumers, this sampling is asynchronous to the data reception, In Link-Triggered consumers, this sampling is done when the data is received.

Data age is defined as the difference between the Consumer\_Clk\_Count at the sample point and the Data Time Stamp when the data was last captured.

#### **2-4.6.4.11     Max\_Data\_Age**

This value indicates maximum value of Data\_Age since the last time that it was reset.

#### **2-4.6.4.12     Consumer\_Fault\_Counter**

For the ExtendedFormat, this a temporary value used to track the number of data integrity errors the consumer detects on the reception of Data and Time Correction messages. This value is incremented every time a Data and Time Correction message is detected with a data integrity error. This value will be reset to 0 every hour. If this value reaches the Max\_Fault\_Number (SafetyOpen Parameter), the connection will be closed.



## **2-5 Safety System Introduction**

This document covers the systems aspect of a Safety Control System that uses a CIP Safety Network for its communications functions. It also defines objects and behaviors that support a safety configuration model (called the SNCT interface) that meets the 61508 requirements with uncertified software and a predominantly non-redundant implementation in the devices. This document will describe those aspects of devices that are necessary to achieve common operations and common behaviors that will make the systems appear seamless to customers using the Safety Network Configuration Tools.

The safety system functionality is organized to easily facilitate moving these requirements into the appropriate CIP protocol documents. Each major section represents one of the major documents; CIP Common, DeviceNet Specification, and the EtherNet/IP Specification.



## **2-6 Safety Connection Establishment**

Establishing safety connections is a key system function. There are two safety elements that need to be addressed; setting up a connection relationship in an originator, and setting up a run-time connection. This section will describe the system requirements to perform both of these operations with integrity. .

### **2-6.1 Configuring Safety Connections**

This chapter defines the parameters needed to perform the safety protocol. It also defines the Safety Segment parameters needed to be passed in the Safety Open to properly set up a safety connection. This section will describe how these parameters get assigned and suggest reasonable defaults for a safety system.

The parameters that need to be configured for a safety connection are:

- Target UNID
- Originator UNID
- EPI
- Ping Interval EPI Multiplier
- Time Coord Msg Min Multiplier
- Timeout\_Multiplier
- Max Consumer Number
- Network Time Expectation Multiplier
- ASYNC
- Max Fault Number (Extended Format only)

All but the ASYNC parameter are included in the Safety Open service.

SRS1 The configuration Software for a Safety Connection Originator shall provide a means for the following parameters to either be configured or given default values: Expected Packet Interval, Target UNID, Originator UNID, Timeout\_Multiplier, Ping Interval EPI Multiplier, Time Coord Msg Min Multiplier, Max Consumer Number, Network Time Expectation Multiplier, ASYNC, Max\_Fault\_Number (Extended Format only). Table 2-6.1 summarizes how these parameters and others will be obtained.



---

Parameter	Suggested Configuration Method	Suggested Default Value
Data Expected Packet Interval (EPI)	User assigned or tool-based estimation	None, user must provide one
Time Coordination EPI	Data EPI * Ping Interval EPI Multiplier	Software Calculated
Time Correction EPI	Data EPI * Ping Interval EPI Multiplier	Software Calculated
Target UNID	User assigned or obtained by software	None, user must provide one
Originator UNID	User assigned	None, user must provide one
Timeout Multiplier	User assigned	2
Ping Interval EPI Multiplier	Default value for inputs or outputs	100 for inputs 19 for outputs
Time Coord Msg Min Multiplier	Default value on path	2 for non-bridged 3 for bridged
Max Consumer Number	Fixed value	15
Network Time Expectation Multiplier	Calculation	Calculated
ASYNCR	EDS File Input	1 (asynchronous)
Initial Time Stamp (Extended Format only)	Dynamic assignment	None, producer must assign a value based on connection format and type
Initial Rollover Count (Extended Format only)	Dynamic assignment	None, producer must assign a value based on connection format and type
Max_Fault Number (Extended Format only)	Default value	5

(in 128uS increments)

$$\begin{aligned} & [\text{EPI\_in\_sec} * \text{Timeout\_Multiplier} \\ & + \text{Safety\_Message\_Time}(\text{max}) \\ & + \text{Time\_Coord\_Message\_Time}(\text{max})] / 128\text{uSec} \\ & + \text{Connection\_Correction\_Constant} \end{aligned}$$

**TABLE 1**

- Safety Message Time (max) = 1 EPI
- Time Coordination Msg Time = Safety Message Time (max)



- Effect of Time\_Drift\_Constant is negligible (Connection Correction Constant can be ignored),

Under these assumptions, the generalized equation can be stated:

Network Time Expectation Multiplier =

$$[EPI\_in\_sec * (Timeout\_Multiplier + 2)] / 128uSec$$

as long as the value does not exceed 5.8 seconds.

The Network Time Expectation Multiplier parameter can now be determined by a common set of user inputs and one EDS file parameter.

Figure 2-6.1 shows where the various parameters shown in Table 2-6.1 are obtained. It also shows how these parameters are used by the software to construct the safety segment of a Safety Open.

**Figure 2-6.1 Sources for Safety Related Connection Parameters**

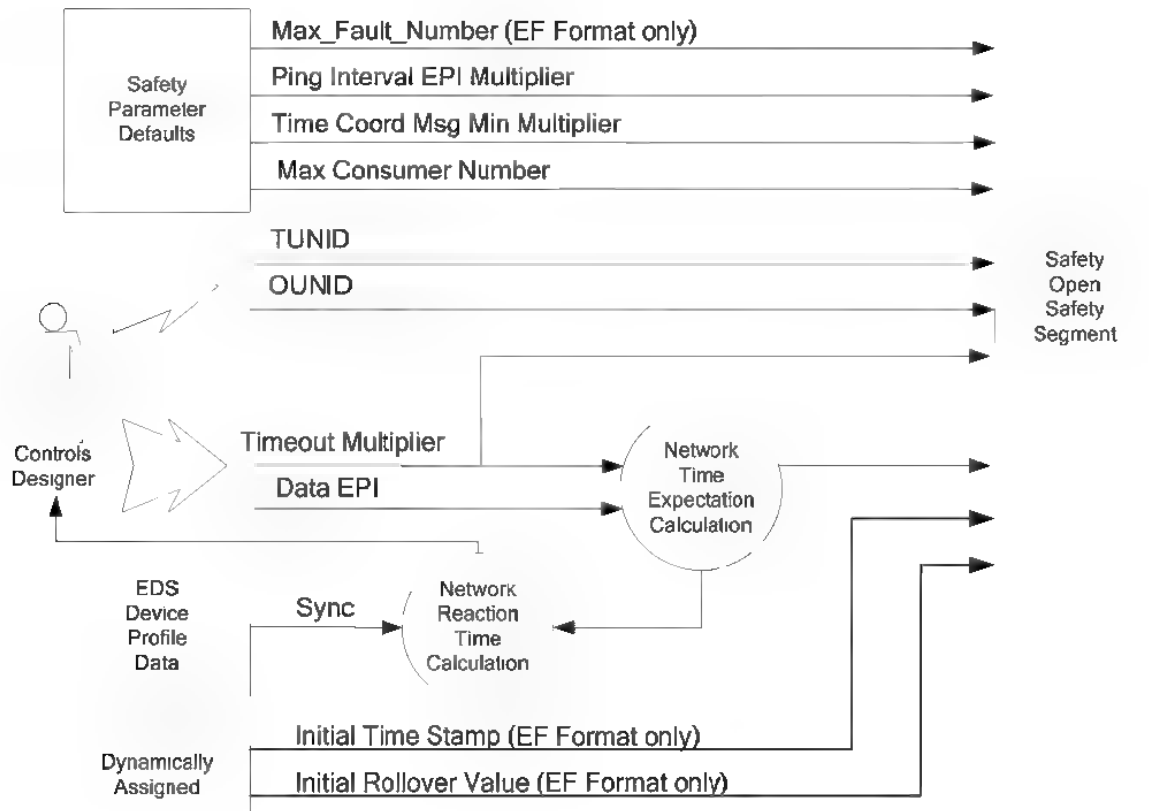
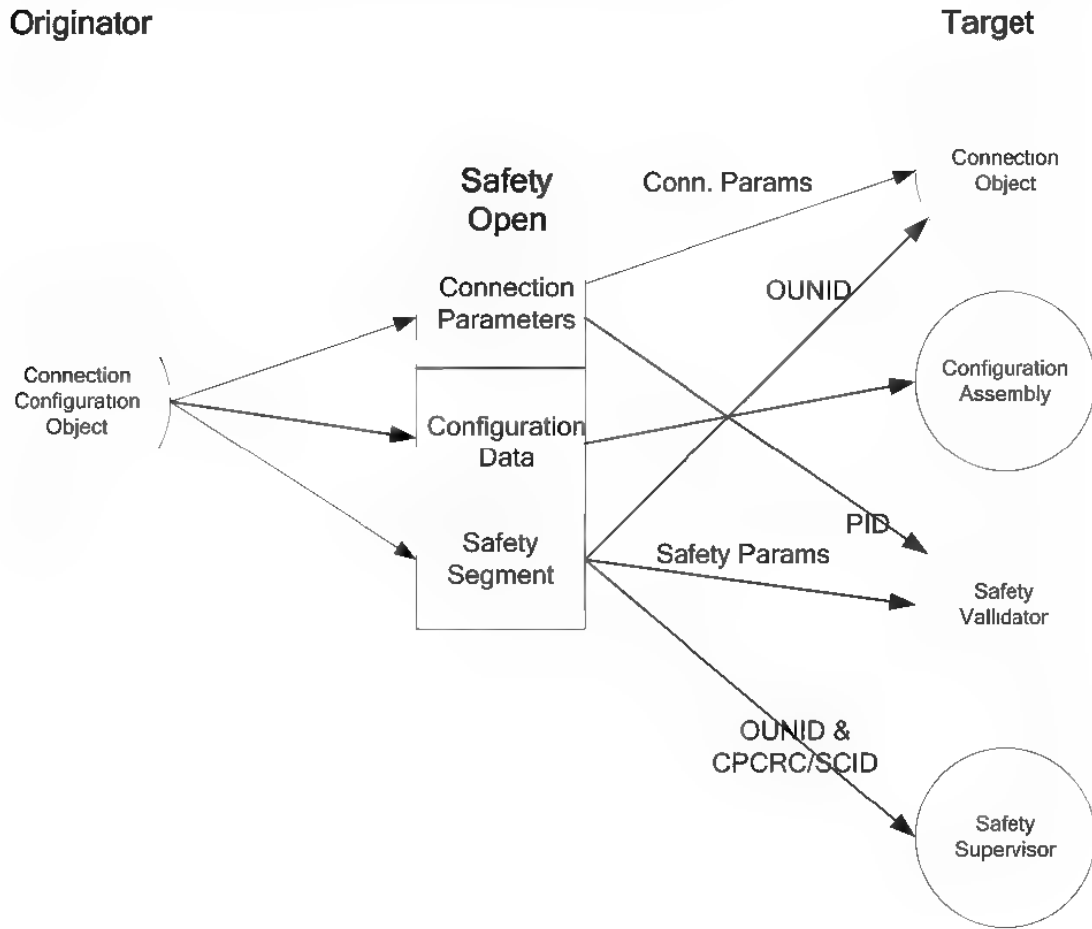


Figure 2-6.2 shows how the Safety Open parameters are mapped in Originators and Targets of a safety connection.



Figure 2-6.2 Parameter Mapping Between Originator and Target





## 2-6.2 Establishing Connections

Section 2-1.9.1.3 defines two types of Safety Open messages. When configuration data is transmitted within the SafetyOpen this is defined as a Type 1 SafetyOpen. When no configuration data is transmitted within the SafetyOpen this is defined as a Type 2 SafetyOpen. This is shown in Figure 2-6.3.

**SRS172** If a connection has configuration data, this configuration data shall only be sent when it is needed. Unless the configuration has been changed, this shall be accomplished by sending a Type 2 SafetyOpen, and only followed with a Type 1 SafetyOpen if the target responds with Device Not Configured error.

**SRS171** When a device receives a Type 1 SafetyOpen containing configuration data and an SCID that matches its existing configuration, it shall re-apply the configuration. All existing rules checks (i.e. OUNID and TUNID checks) shall still apply.

Since all safety nodes are required to have NV storage for configuration data, this is only likely on device replacement, initial start-up or when the configuration has changed. This process is illustrated in Figure 2-6.4.

**SRS173** Targets shall respond to a Type 2 SafetyOpen with an error code 0x01, additional code 0x0110 (device not configured), when the device is not configured.

**SRS174** When an originator detects that the configuration data held for a device has changed, it shall always issue a type 1 connection request the first time it connects. The SCID returned shall be validated against the value held.

There are two kinds of Type 2 Safety Opens: with and without the SCID

**SRS2** Type 2a: When a target receives a Type 2 SafetyOpen (no configuration data) accompanied by a non-zero Safety Configuration ID (SCID = SCCRC+SCTS), it shall compare it against the SCID of the configuration and only accept the connection if they match.

Type 2b: When a target receives a Type 2 SafetyOpen (no configuration data) accompanied by a zero value SCID, it skips the check (refer to FRS163).

**Table 2-6.2 SafetyOpen Summary**

	Type 1 SafetyOpen With Data	Type 2a SafetyOpen with SCID check	Type 2b SafetyOpen without SCID check (FRS163)
TUNID	X	X	X
OUNID	X	X	X
UPID <sup>1</sup>	X	X	X
CPCRC	X	X	X
SCID	X	X	= 0
Config. Data in Safety Open	Yes	No	No

<sup>1</sup>UPID is derived from components in the Safety Open or from application data returned in the SafetyOpen response. Refer to Section 2-6.7



The processing of SafetyOpens in Targets will differ depending on the type of SafetyOpen message that is received. Figure 2-6.4 shows the message and event sequence for the Type 1 Safety Open and the objects that are responsible for the various processing steps. Figure 2-6.3 shows that the primary difference between processing Type 1 and Type 2 is the step where configuration data is passed to a configuration assembly and an owner assigned to that configuration.

Figure 2-6.3 DeviceNet Connection Establishment in Targets for Type 2a Safety Open

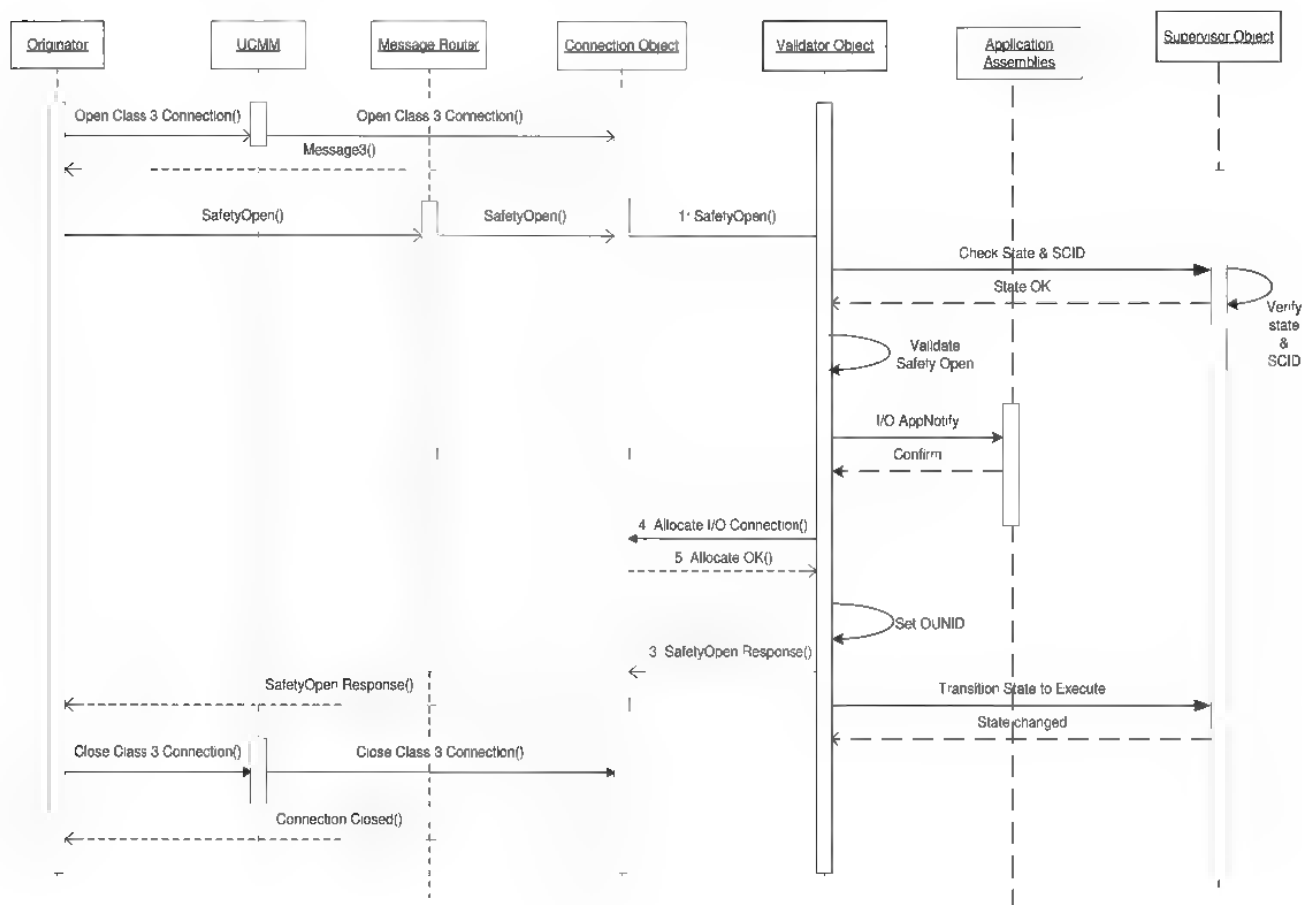
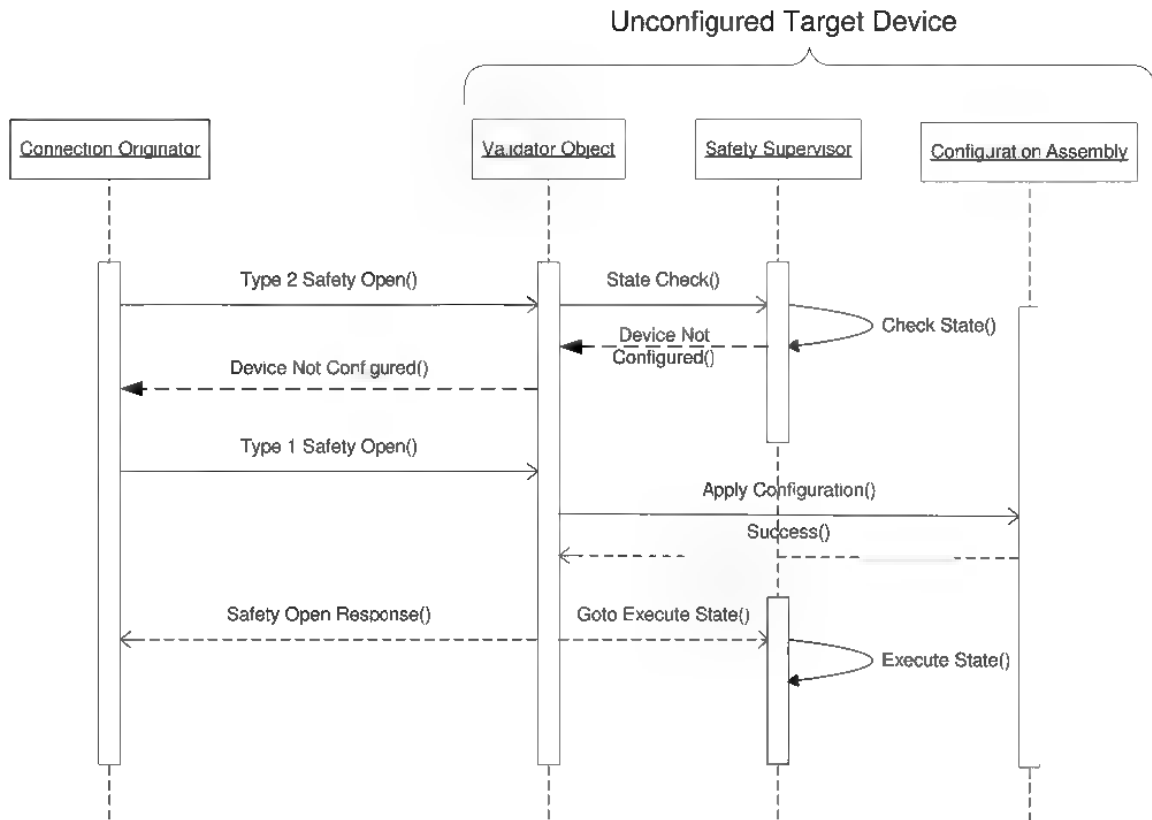




Figure 2-6.4 General Sequence to Detect Configuration is required





### **2-6.3 Recommendations for Consumer Number Allocation**

When a Multi-cast connection is established, the producer of that connection must allocate a unique consumer number to the consumer. The main difficulty with this process is not in the initial allocation, but in dealing with the case where a producer has closed a connection for some reason and now needs to decide when it is acceptable to re-assign that consumer number again.

Since multi-cast producers will continue to produce data even though a connection to a particular consumer was closed (i.e. assumes other consumer connections still open), the only mechanism available for a multi-cast consumer to detect the loss of the connection is by detecting that it has not gotten a Time Correction message within  $\text{Timeout\_Multiplier.PI}+1$  ping intervals. During this period, the Consumer still thinks the connection is open, is still processing received data, and is still sending Time Coordination responses (even though the producer ignores them).

If a producer were to get a new connection request from another consumer and re-allocate this consumer number, there would be two consumers using the same number during this period. The producer would again start processing Time Coordination responses and sending Time Correction messages with that consumer number. In other words, if the timing were just right, the old consumer might never detect the connection has failed and there would exist two consumers with the same number sending conflicting clock information and using conflicting correction values. While this case will ultimately be detected by the producer through PID seeding, a means shall be employed to minimize the occurrence.

Multi-cast producers shall quarantine consumer numbers that have been taken off-line for  $\text{Timeout\_Multiplier.PI}+2$  ping intervals before it can be re-allocated to another connection. The only exception to this is when a Consumer issues a `Forward_close` to notify the producer that the connection is being closed. In this case, the consumer number can be re-used immediately.

### **2-6.4 Recommendations for Connection Establishment**

When a multi-cast consumer decides to unilaterally close a connection, the only way for a producer to detect this event (other than getting a `Forward Close` from the consumer) is when it no longer receives a Time Coordination response for  $\text{Timeout\_Multiplier.PI}+2$  Ping Intervals. During that time interval, the consumer number is still allocated to that consumer. There is a strong likelihood that while this connection is still open, that consumer will issue a new `SafetyOpen`. Producers have two options in this case:

- Search through the existing connections and see if a connection to this same consumer is already established and if so, re-issue the consumer number again.
- Allocate a new consumer and allow the old consumer number to time-out and go through quarantine.

The first option is the more complicated to code, but is the recommended option since it offers a more robust re-allocation scheme for consumer numbers. The advantage of the second option is that it is simpler to implement, but allows for consumer numbers to languish in quarantine longer than necessary. The second option can also make connection startup problematic for applications where the number of consumers is  $> 8$ .



If option one is used, the recommended scheme to match up connections should be to match up:

- OUNID and
- Application data path

If these match up with the parameters of an existing connection, then that device's consumer number can be re-issued.

## **2-6.5 Ownership Establishment**

Section 2-1.9 discussed two methods of establishing connection ownership, through the software tools or from the SafetyOpen. It indicated that devices are required to support a facility to allow "tool only" configuration or originator configuration. It also implied that attempts to reconfigure a device via the SafetyOpen will be rejected if the configuration is locked. If the "tool only" state is not set, the OUNID of the selected originator can also be set by the software tool during target device commissioning for added security.

This section will define how this capability has been defined in the new Safety Objects. The new Safety Objects are required in devices that will be configured by the Safety Network Configuration Tool (SNCT).

To establish ownership of the configuration (and the connection in outputs), the OUNID of the originating device is passed to the target in the SafetyOpen message.

SRS4 Safety Devices shall NV-store the OUNID for the configuration in the device. SRS5 Also, Safety Devices shall NV-store the signature for the configuration (SCID) in the device. The SCID is derived by concatenating the SCCRC and the SCTS in the SafetyOpen.

SRS6 Output devices shall NV-store the OUNID associated with the safety output connection.

When the user has tested and verified the tool-generated configuration, they can lock the configuration from changes and flag it as "Locked". The "Configuration Lock" flag is an attribute of the Safety Supervisor. To modify it, a password can be provided which provides user defined security.

As a side note, devices which are configured by the SafetyOpen cannot have a "Configuration Lock" condition. The assumption is the Lock condition is set and controlled at the connection originator.

## **2-6.6 Ownership Use Cases**

This section will present a series of connection scenarios where configuration is also being attempted. Each scenario will provide a brief description of how the device should deal with each case. Most of these cases utilize the Safety Open to configure the device. As such, the tool-based configuration features such as the "Configuration Lock" attribute must not be in use.



**1. The user-designated owner connects and configures an un-owned, un-configured input device – (OUNID assigned at first connection)**

The Configuration Lock attribute in the target must be “unlocked” and the Safety Supervisor must be in Configure mode; device has no configuration. The SafetyOpen is validated by the target device as being correct and the originator OUNID is stored in the target and related to the configuration. A single-cast or multi-cast connection is established. Normal operation starts.

SRS7 The originator of a Type 1 SafetyOpen that configures a previously unconfigured & unowned input device shall become the owner of the configuration.

**2. The user-designated owner connects and configures an un-owned, un-configured output device – (OUNID assigned at first connection)**

The Configuration Locked attribute in the target must be “unlocked” and the Safety Supervisor must be in Configure mode; device has no configuration. The SafetyOpen is validated by the target device as being correct and the OUNID is stored in the target and related to the configuration **and the connection**. A single-cast connection is established. Normal operation starts.

SRS8 A Type 1 SafetyOpen that configures a previously unconfigured & unowned output device shall become the owner of both the connection and the configuration .

**3. Owner connects and configures an owned, un-configured input device – (OUNID assigned by tool)**

The Configuration Lock attribute in the target must be “unlocked” and the Safety Supervisor must be in Configure mode; device has no configuration. The SafetyOpen is validated by the target device as being correct and the SafetyOpen OUNID matches the stored OUNID. A single-cast or multi-cast connection is established. Normal operation starts.

**4. Owner connects and configures an owned, unconfigured output device – (OUNID assigned by tool)**

The Configuration Lock attribute in the target must be “unlocked” and the Safety Supervisor must be in Configure mode; device has no configuration. The SafetyOpen is validated by the target device as being correct and the SafetyOpen OUNID matches the stored OUNID for the configuration **and the connection**. The target stores the configuration. A single-cast connection is established. Normal operation starts.

A second device attempts to connect to an owned single-cast output connection

If no connections are open the SafetyOpen will be evaluated by the target device. SRS9 If a Type 1 SafetyOpen is sent where the originator OUNID is different than the OUNID that is stored with the connection the SafetyOpen shall be rejected and an error message, “O\_UNID Mismatch” is returned

**5. Owner connects and attempts to re-configure a tool-owned, configured device – (Configuration Lock set)**

SRS10 If an originator attempts to configure a device with the Configuration Lock attribute set, the device shall reject the SafetyOpen and return an error message “Configuration Not Allowed” is returned



## **6. Device reconfiguration – Input Device**

The owner of a device sends a SafetyOpen with configuration data plus an SCID (SCCRC & SCTS) to the target device. The data may or may not be the same as what exists in the device and the SCID may or may not match the existing SCID. Regardless, as long as the following process is followed, it should treat the message as a re-configuration.

SRS11 A target input device that has an existing configuration shall (at a minimum) do the following when a type 1 Safety Open is received:

- Verify that the TUNID in the SafetyOpen matches the device TUNID
- If configuration data is included (type 1), verify that the OUNID in the SafetyOpen matches the stored OUNID
- Verify the Electronic Key (See THE CIP NETWORKS LIBRARY, Volume 1, Common Industrial Protocol (CIP)) matches the device
- Calculate the SCCRC over the received data and confirm that it obtains the same value
- verify the CPCRC over the configuration is correct,
- start its reconfiguration process (enters Config State), close and inhibit connections during the reconfiguration (respond to connection requests with an error response, “Invalid Mode/State”),
- update the configuration and SCID in NV Memory,
- verify and validate the connection parameters,
- Validate the application path
- Confirm the safety connection requested is supported
- Safety parameters are within valid ranges
- transition the connection to the established state, and the device to the executing state.
- New connections can now be accepted

A connection of the requested type is established. While new connections can again be accepted, these other devices must now either have the new SCID, or connect with an SCID = 0.

## **7. Device reconfiguration – Output Device**

The owner of a device sends a SafetyOpen with configuration plus an SCID to the target device. The data may or may not be the same as what exists in the device and the SCID may or may not match the existing SCID. Regardless, as long as the following process is followed, it should treat the message as a re-configuration

SRS12 A target output device that has an existing configuration shall (at a minimum) do the following when a type 1 Safety Open is received.

- If the connection path is to an output resource, verifies that the OUNID in the SafetyOpen matches the OUNID for the connection,
- Verify that the TUNID in the SafetyOpen matches the device TUNID
- Verify the Electronic Key (See THE CIP NETWORKS LIBRARY, Volume 1, Common Industrial Protocol (CIP)) matches the device
- If configuration data is included (type 1), verify that the OUNID in the SafetyOpen matches the stored OUNID for the configuration
- Calculate the SCCRC over the received data and confirm that it obtains the same value
- verify the CPCRC over the configuration is correct,



- start its reconfiguration process by closing and inhibiting connections during the reconfiguration (respond to connection requests with an error response, “Invalid Mode/State”),
- update the configuration and SCID in NV memory,
- verify and validate the connection parameters,
- Validate the application path
- Confirm the safety connection requested is supported
- Safety parameters are within valid ranges
- transition the connection to the established state, and the device to the executing state.

A connection of the requested type should now be established.

#### **8. Connection establishment to a previously owned device (Changing OUNID)**

When attempting to configure a device that has another owner, a SafetyOpen configuration request (i.e., Type 1) without a matching OUNID will be rejected with an error response, “OUNID Mismatch”.

SRS129 To clear an existing OUNID, safety targets shall support a “reset to out-of-box” using a reset switch on the device or using the special reset command defined in the Safety Supervisor.

SRS15 If the device has a Safety I/O connection when a reset command is received, the reset command shall be rejected and an error message returned, “Invalid Mode/State”. In order for a device to be reset by command it must not have Safety I/O connections established.

A second device attempts to connect to an owned multi-cast connection

A second originator sends a SafetyOpen (i.e., Type 2a, or Type 2b) to a device that has a multi-cast connection owned by another originator. If the SCID in the Safety Open is non-zero, the SCID is compared to the saved SCID. If the signatures match, the connection is established. If the signature does not match, the error response “SCID Mismatch” is returned. The owner does not need to be active during this process.

SRS180 A target device that has an existing configuration shall (at a minimum) do the following when a type 2a or 2b Safety Open is received.

- If the connection path is to an output resource, verifies that the OUNID in the SafetyOpen matches the OUNID for the connection,
- If the SCID is non-zero, confirm the SCID matches the device SCID
- Verify that the TUNID in the SafetyOpen matches the device TUNID
- Verify the Electronic Key (See THE CIP NETWORKS LIBRARY, Volume 1, Common Industrial Protocol (CIP)) matches the device
- verify the CPCRC over the configuration is correct,
- verify and validate the connection parameters,
- Validate the application path
- Confirm the safety connection requested is supported
- Safety parameters are within valid ranges
- transition the connection to the established state



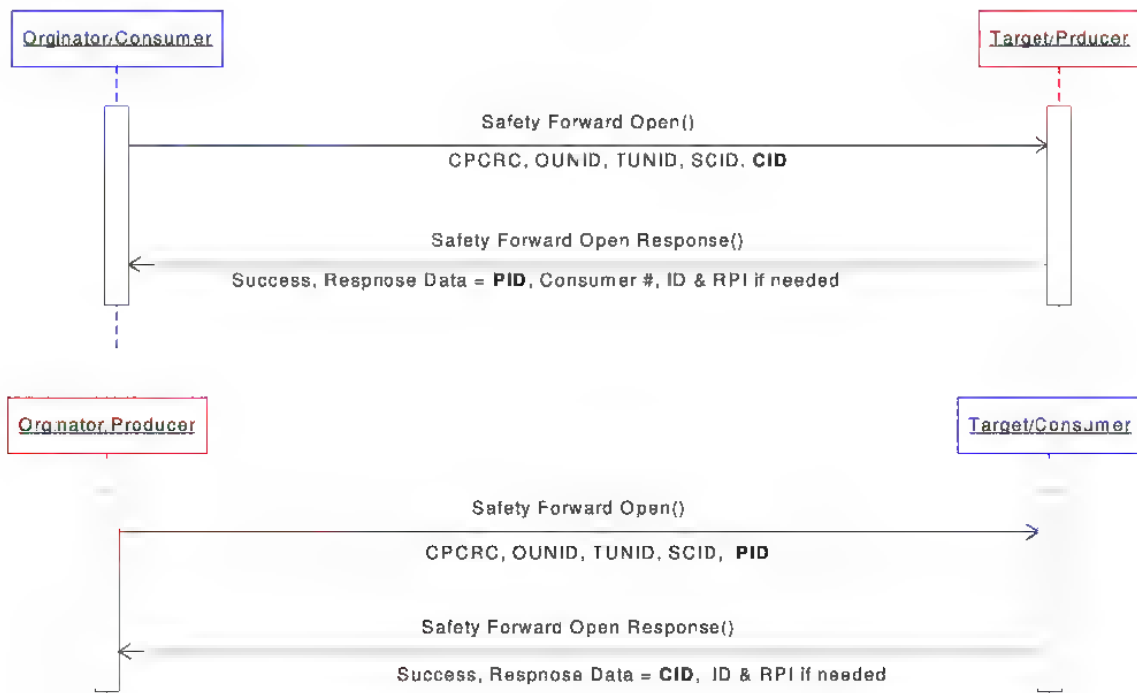
## 2-6.7 PID/CID Usage and Establishment

The PID/CID seeding mechanism has been defined to detect message insertion from an invalid source. This insertion can occur from another producer sending Safety Data, another producer sending Time Correction messages, or an invalid consumer sending Time Coordination messages. All these cases will be covered by the PID/CID scheme in the Safety Protocol. The key to this is the exchange of PID/CID information during the connection establishment process. The Producer will provide its PID to all its consumers, and all the consumers shall provide their respective CID back to the producer. This exchange occurs regardless of who originates the connection, or what type of safety connection is being established.

SRS152 During connection establishment the producing and consuming nodes shall exchange their respective PID/CID information (Vendor ID + Device Serial Number + Connection Serial Number). The originator sends its ID in the SafetyOpen request, and the target returns its ID as part of the SafetyOpen Success response. The two cases are shown in Figure 2-6.5.

SRS178 The PID shall be used to seed Data production CRCs and Time Correction CRC, and the CID shall be used to seed the Time Coordination CRC.

Figure 2-6.5 PID/CID Exchanges for Two Originator Scenarios



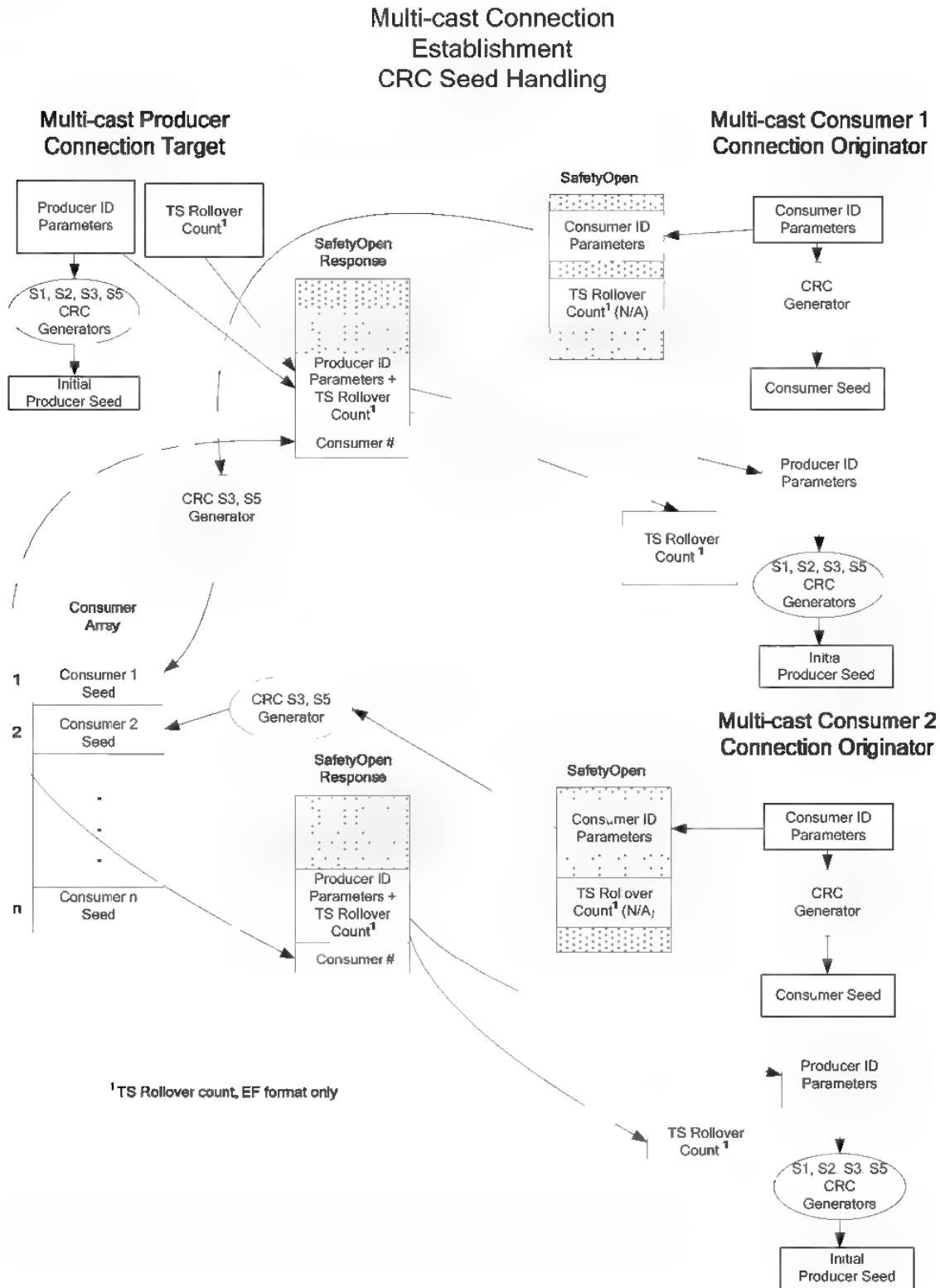


### **2-6.7.1 Proper PID/CID Usage in Multi-cast & Single-Cast Connections**

The diagram shown in Figure 2-6.6 illustrates how the Producer and Consumer ID parameters are to be used to generate the CRC seed values for both base and Extended Format connections. This diagram shows the multi-cast case, but it also applies to Single-cast connections as well (this is done simply for consistency). References to TS\_Rollover\_Cnt and Rollover Detector only applies to the Extended Format is not used in the base formats. The producer seeds are generated using only the S1, S3 and S5 generators (i.e. same seed used for S1 and S2 CRC calculations) and the CID seed is generated with the S3 and S5 generator. For more information, refer to Section 2-1.7.1.



Figure 2-6.6 Seed Generation for Multi-cast Connections





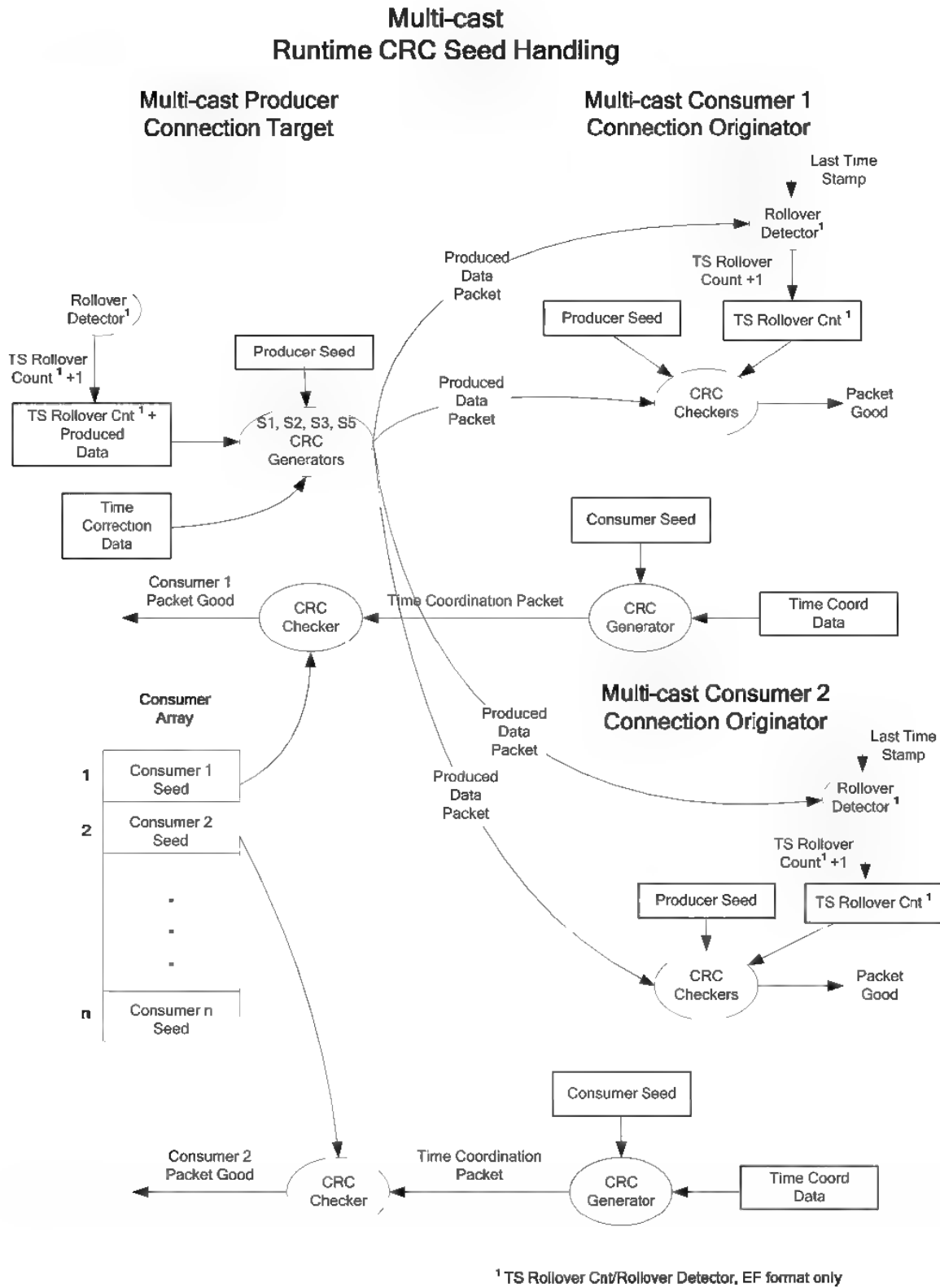
PID parameters shall be used to seed the CRCs for Produced Data messages and Time Correction messages, and CID parameters shall be used to seed the CRC for the Time Coordination responses.

As shown in the figure, multi-cast producers will need to maintain a consumer array to keep track of the seed value for each allocated consumer number. The scheme shall also apply to Single-cast connections except that only one consumer seed value needs to be maintained.

The diagram shown in Figure 2-6.7 illustrates how the seed values, set up during connection establishment, get used in the run-time data production.



Figure 2-6.7 PID/CID Runtime Handling





## **2-6.8 Network Supported Services**

The purpose of this section is to promote compatibility between devices developed to exchange safety data over the CIP Safety Networks. It defines guidelines on which safety services should be supported by the different CIP Safety Device Types. The goal is to have basic services that are supported by all devices that intend to support a particular CIP Safety Connection Category.

The goal is to allow the development of input and output devices which range from very simple discrete I/O to smart high functionality specialty I/O modules. When an input or output module reach a certain level of functional requirements, they may need to be viewed as embedding a safety Logic Device within the module.

### **2-6.8.1 CIP Safety Device Type**

To simplify the exchange of information and presentation of Safety Data to the users of multiple systems, the devices that participate on the CIP Safety Network are divided into 4 device types.

- Input Devices
- Logic Devices
- Output Devices
- I/O Devices

Input Devices are devices that translate user signals into safety data and send the safety data to Logic Devices. Input Devices do not receive safety data other than the safety connection status. If a device with input information needs to receive safety data from the other side of the safety connection, it should be considered an I/O Device or a Logic Device, and follow the guidelines of the I/O or Logic Safety Device Type.

Logic Devices are devices that process Safety Input Data and Produce Output Data. Direct communication between Input Devices and Output Devices is not supported. A Logic Device, no matter how simple, has to be between an Input Device and an Output Device.

Output Devices are devices that present some signal to the user's environment based on the safety data consumed by the device. Output Devices may have status that is produced back to the Logic Device.

I/O Devices are devices that have both Safety Inputs and Safety Outputs.

### **2-6.8.2 CIP Safety Connection Category**

All Safety Connection can be placed in one of four Categories. They are:

- Input\_Device\_To\_Logic\_Device
- Logic\_Device\_To\_Output\_Device
- Logic\_Device\_To\_I/O\_Device
- (an Output device with Status is considered an I/O Device)
- Logic\_Device\_To\_Logic\_Device



Logic Devices may chose to support any or all of the Safety Connection Categories. If a device does support a Safety Connection Category, it should support all the Recommended Services of that category.

### 2-6.8.3 Safety Connection Services

The following table lists all the defined Safety Validator Services that Connection Targets or Connection Originators can support.

**Table 2-6.3 Originator/Target Service Mapping**

Connection Originator or Target	Safety Connection Types		Validator Type
Target	Single-Cast	Producer	Type 1
Target	Single-Cast	Consumer	Type 2
Target	Multi-Cast	Producer	Type 3
Originator	Single-Cast	Producer	Type 1
Originator	Single-Cast	Consumer	Type 2
Originator	Multi-Cast	Consumer	Type 4

The following table lists the Safety Validator Services that Connection Targets or Connection Originators should not support. Multi-cast producers are not required to initiate connections to its consumers; the consumers must connect to it; Multi-cast Consumers should be the connection Originator, and Multi-cast Producers should be connection targets.

**Table 2-6.4 Unsupported Originator/Target Service Types**

Connection Originator or Target	Safety Connection Types		Validator Type
Target	Multi-Cast	Consumer	N/A
Originator	Multi-Cast	Producer	N/A

### 2-6.8.4 Services Supported for each Category

The following tables list the preferred Safety Connection Services for various Connection Categories. For each Safety Connection Category, the different columns represent preferred methods of establishing safety connections between the two devices. For any particular device, only one column would be supported at one time.



**Table 2-6.5 Connection Categories and Supported Services**

					Supported Services for Input or Output Type						
Safety Connection Category					1		2	3			
Safety Connection Service					Input Device To Logic Device		Logic Device To Output Device	Logic Device To I/O Device (Output w Status)			
					Originator or Target	Safety Connection Type			Op 1	Op 2	
Input & Output Devices	Target	Single-Cast	Producer			O			O		
	Target	Single-Cast	Consumer				R	R	O		
	Target	Multi-Cast	Producer		R			R			
Logic Devices	Originator	Single-Cast	Producer				R	R	O		
	Originator	Single-Cast	Consumer			O			O		
	Originator	Multi-Cast	Consumer		R			R			

Op1,2,3 = Optional Combinations

R = Recommended Services for compatibility

O = Optional Service



Figure 2-6.8 Recommended Connection Types

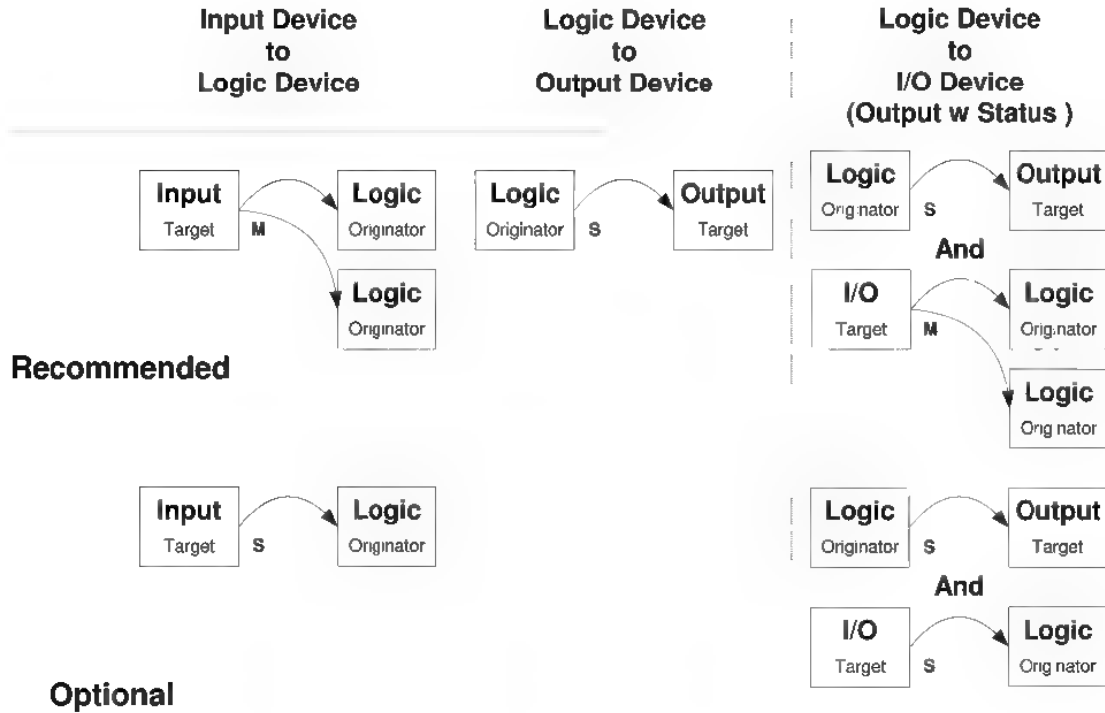


Table 2-6.6 Logic-to-Logic Supported Services

	Safety Connection Service			Safety Connection Category 4 Logic Device to Logic Device			
	Originator or Target	Safety Validator Type		Supported Service Combinations			
				Op 1	Op 2	Op 3	Op 4
Logic Device Target	Target	Single-Cast	Producer			O	
	Target	Single-Cast	Consumer		O		
	Target	Multi-Cast	Producer	R			
Logic Device Originator	Originator	Single-Cast	Producer		O		
	Originator	Single-Cast	Consumer			O	
	Originator	Multi-Cast	Consumer	R			

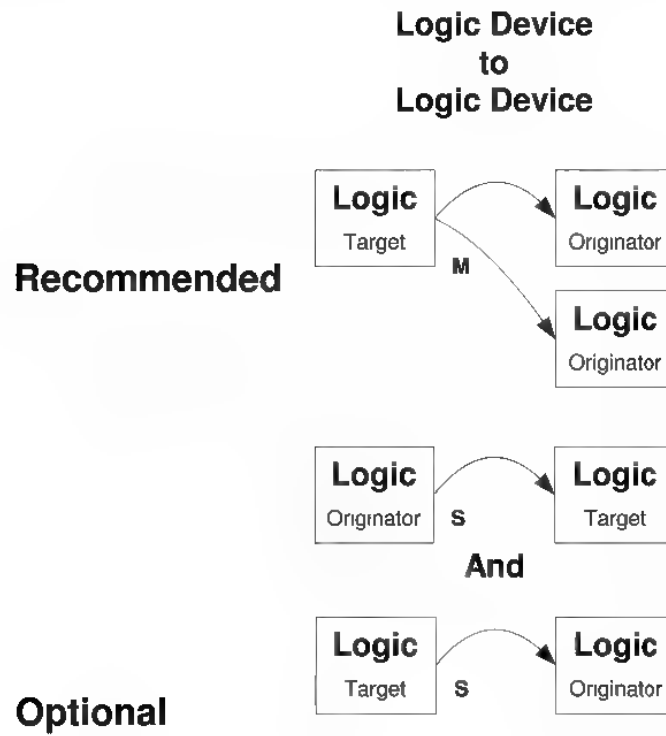
s= Single Cast, m= Multi-cast,

R=Recommended Basic Service for compatibility,

O=Optional Service



Figure 2-6.9 Recommended Connection Types for Logic to Logic



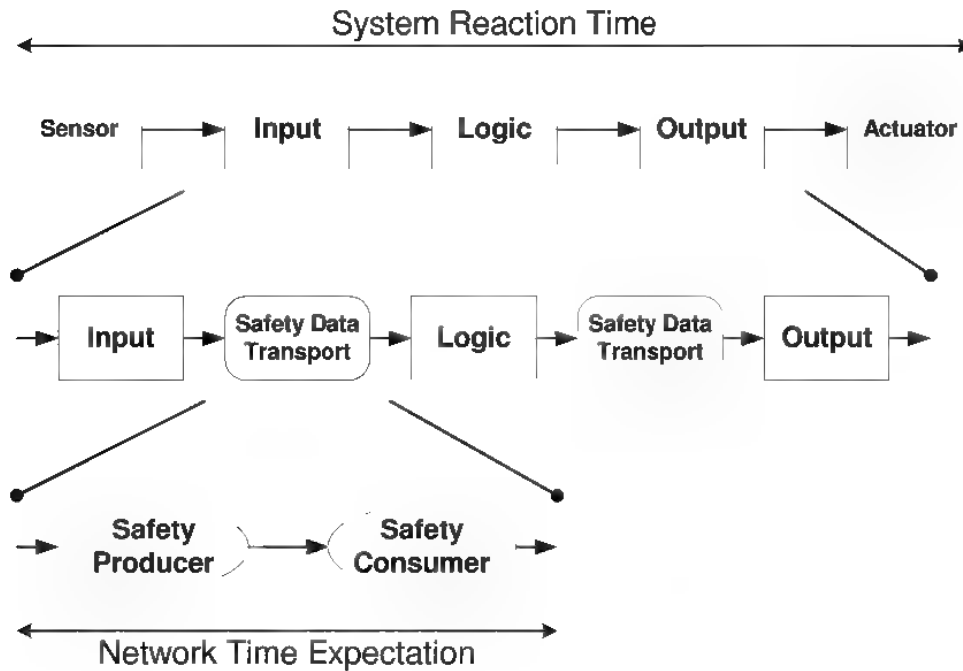


## 2-7 System Reaction Time

### 2-7.1 Introduction

The system reaction time is the worst-case time from a safety related event as input to the system or as a fault within the system, until the time that the system is in the safety state.

Figure 2-7.1 System Reaction Time



To determine the system reaction time of any control chain the user shall add up the components of the safety chain.

For example, the system reaction time of the example above would be:

System reaction time = Sensor reaction time  
 + Input reaction time  
 + Network reaction time  
 + Controller reaction time  
 + Network reaction time  
 + Output reaction time  
 + Actuator reaction time

### 2-7.2 Network Time Expectation

The Network Time Expectation is a portion of the System Reaction Time. The Network Time Expectation is the worst case time, from the time the data is captured by the safety data producer, until the consuming application recognizes a safety state. This also includes errors during production and consumption.



The **Network\_Time\_Expectation\_Multiplier** used in the previous sections of this document, represents the worst case measured time from the time the data is captured by the safety data producer until the time that the consuming application recognizes a safety state indication.

The **Network\_Time\_Expectation\_Multiplier** value necessary to ensure a timeout error is not detected is:

$$\begin{aligned} \text{Network\_Time\_Expectation\_Multiplier} > \\ & (\text{EPI} * \text{Timeout\_Multiplier} \\ & + \text{Safety\_Message\_Time}(\text{max}) \\ & + \text{Time\_Coord\_Message\_Time}(\text{max})) / 128 \mu\text{Sec} \\ & + \text{Connection\_Correction\_Constant} \end{aligned}$$

Where:

**Safety\_Message\_Time(max)**, is the actual time from the data being captured by the safety data producer until the time that the safety data is passed to the consuming application for use.

**Time\_Coord\_Message\_Time(max)**, is the maximum time it could take for the time coordination information to be sent from the consumer to the producer. This is measured from the time the **Consumer\_Clk\_Count** is captured to **Consumer\_Time\_Value** until the time **Producer\_Clk\_Count** is captured to **Producer\_Rcvd\_Time\_Value**.

**Timeout\_Multiplier** is a parameter that is needed by the CIP safety protocol processing (refer to FRS230). This value determines the number of messages that may be lost before declaring a connection error. A **Timeout\_Multiplier** of 1 indicates that no messages may be lost. A **Timeout\_Multiplier** of 2 indicates that 1 message may be lost, etc. In addition to the **Timeout\_Multiplier** checks within the protocol, reaction time checks are also performed. A reasonable default for the **Timeout\_Multiplier** would be 2. A reasonable limit would be for configuration software to restrict the **Timeout\_Multiplier** to integer values from 1 to 4. For the **ExtendedFormat** the **Timeout\_Multiplier** may be increased to 255 as long as the **Network Time Expectation** value does not exceed 5.8 seconds.

**Connection\_Correction\_Constant** is a value in 128  $\mu\text{Sec}$  increments that is subtracted from the time stamp to represent the worst case error due to Time drift, the asynchronous nature of the producer and consumer clocks, and the minimum time for the Time Coordination Message to traverse from the consumer to the producer.

See the Section 2-6.1 for a discussion of the simplified forms of this equation that can be used for configuration purposes.

### 2-7.3 Produced Data Network Reaction Time

For CIP safety networks used on producing devices that have a constant delay or whose safety time is synchronized to the data production:

$$\text{Network Reaction Time} = \text{Network\_Time\_Expectation\_Multiplier} * 128 \mu\text{Sec}$$

For CIP safety networks used on producing devices that have a producing application periodic cycle not synchronized to the data production:

$$\text{Network Reaction Time} = \text{Network\_Time\_Expectation\_Multiplier} * 128 \mu\text{Sec} + \text{EPI}$$



## 2-7.4 Equations for Calculating Network Reaction Times

The most logical place for the user to set and determine the Network reaction times is at the logic device since it is very likely both the configuration holder and connection originator.

There are two cases of safety connection reaction times for most logic devices. They are:

- Input Connection reaction time
- Output Connection reaction time

The following table defines the producing application and consuming application for the Network reaction time Cases above.

**Table 2-7.1 Connection Reaction Time Type – Producing/Consuming Applications**

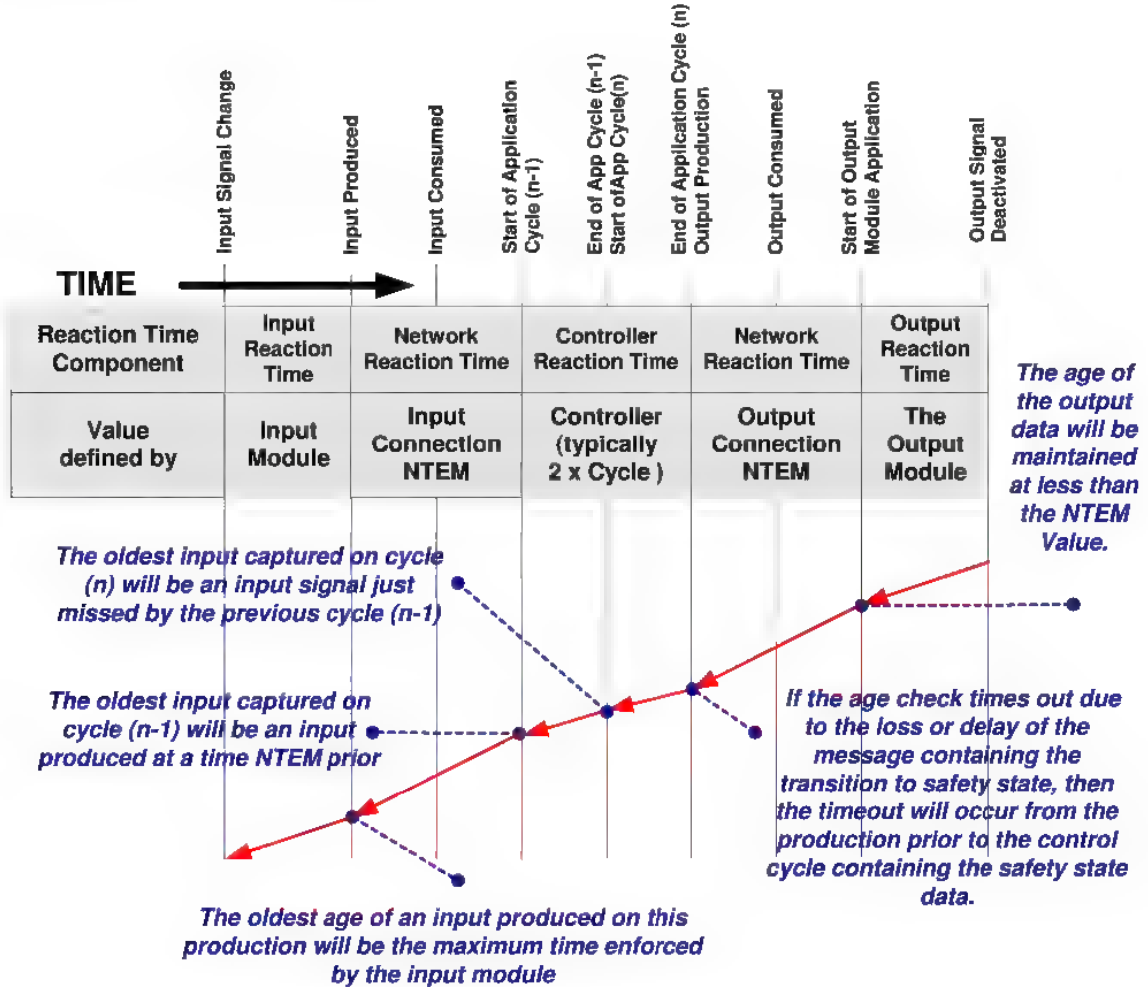
Connection Reaction Time Type	Producing Application	Consuming Application
Safety Input	Input Module	Controller
Safety Output	Controller	Output Module

Figure 2-7.2 shows the analysis of the Reaction Time components. The time enforcement analysis is done from the Output signal back to the Input signal. Doing the analysis from the Output signal back to the Input signal allows the effects of asynchronous productions to be seen.

The analysis assumes that the output module will have an internal cycle that checks the age of the last received output data. The age of the output data will be maintained at less than the Network\_Time\_Expectation\_Multiplier (NTEM) value.



Figure 2-7.2 Reaction Time Derivation



From an analysis point of view, a controller to controller connection would be treated the same as an Input connection.

For an Input or Controller to Controller Connection:

**Network reaction time = Network\_Time\_Expectation\_Multiplier \* 128µS**

For a Connection to an Output Module:

If the Output Production is Asynchronous to the Controller Cycle:

**Network reaction time = Network\_Time\_Expectation\_Multiplier \* 128µS**

If the Output Production is Synchronous to the Controller Cycle:

**Network reaction time = Network Time Expectation Multiplier \* 128µS - EPI in Sec**

The Output Connection cases can be simplified to:

**Network reaction time = Network Time Expectation Multiplier \* 128µS + (EPI\_in\_Sec \* (ASYNC - 1))**

Where the ASYNC parameter is defined as follows:



## ASYNCR Parameter

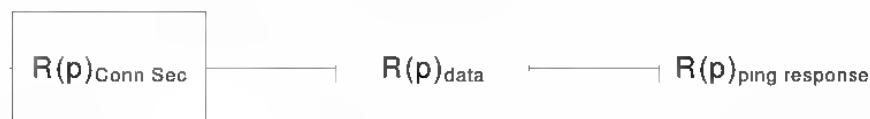
If the producing application is synchronous with data production, ASYNCR=0. If the producing application is asynchronous to the data production, ASYNCR = 1. This ASYNCR parameter shall be provided to the Safety Configuration software via an EDS file entry. The default value in cases where it is not provided shall be 0.

## 2-8 Network PFH

This section will describe the calculations used for the determination of the network PFH. The network PFH must meet the 1% rule from the German Safety Bus Paper<sup>4</sup>. The network is targeted at SIL 3 systems therefore the PFH must be <10<sup>-9</sup> (or <10<sup>-7</sup> if a 1% factor is included in the calculations).

The safety network protocol consists of four parts, safety data, time stamp information, the multi-cast time correction information, and the time coordination ping response. Each of these parts has its own protection that contributes to the PFH.

Figure 2-8.1 Network Protocol Reliability Block Diagram (RBD)



From the RBD in Figure 2-8.1, the equation for the overall network residual error rate is:

### Equation 2-8.1 Overall Network Residual Error Rates

$$R(p)T = R(p)_{data} + R(p)_{time\ stamp} + R(p)_{time\ coordination} + R(p)_{time\ correction}$$

When applying the residual error rates in the calculation of the PFH, it is important to note that the time coordination and the multi-cast time correction information are transmitted at rates other than the rate of the time stamp and data. Using the different transmission rates and the information from the RBD, the following equation for the overall PFH can be derived:

### Equation 2-8.2 PFH for Single-Cast Connections

$$\Lambda = 3600 * (((R(p)_{time\ stamp} + R(p)_{data}) * v_{messages}) + (R(p)_{time\ coordination} * v_{ping})) * (Num\ of\ Nodes - 1) * 100$$

The equation for multi-cast PFH is shown below:

<sup>4</sup> The one percent factor is used to assure that the network will only use 1% of the PFH budget, based on the recommendations of the German Safety Bus Guidelines. See: Draft proposal test and certification guideline, safety bus systems, BG Fachausschuß Elektrotechnik 28-May-2000,



**Equation 2-8.3 PFH for Multi-cast Message Connections**

$$\Lambda = 3600 * (((R(p)_{\text{time stamp}} + R(p)_{\text{data}}) * v_{\text{messages}}) + ((R(p)_{\text{time coordination}} + R(p)_{\text{time correction}}) * v_{\text{ping}} * \text{NoC/P})) * (\text{Num\_of\_nodes} - 1) * 100$$

The factors in the above equations are:

vmessages	Transmission rate in messages/second
vping	Transmission rate in messages/second
NoC/P	Number of Consumers per Producer
3600	conversion of hours to seconds
100	1% factor

The following table summarizes the worst-case result. The numbers are obtained from detailed calculations and the formulas above.

**Figure 2-8.2 Summary of PFH Calculations**

<b>Inputs</b>			
	Expected Packet Interval (EPI)	0.010	Seconds
	Ping Interval	0.524	Seconds
	Number of Nodes (m)	63	
	Multi-Cast Consumers	15	
	Bit Error Rate (p)	0.001	
<b>Results</b>			
		<b>Single Cast</b>	<b>Multi Cast</b>
R(p)data <= 2 bytes	R(p) data section - 2 bytes of data	7.02E-20	7.02E-20
R(p)data >= 3 bytes	R(p) data section - 250 bytes of data	2.85E-24	2.85E-24
R(p)data	R(p) data section	7.02E-20	7.02E-20
R(p)ts	R(p)time stamp section	3.18E-18	3.18E-18
R(p)multi	R(p) Time correction message		1.40E-18
R(p)reply	R(p) Time coordination message	1.40E-18	1.40E-18
Msg_per_sec	Messages per second = 1 / EPI	100	100
Rpl_per_sec	Replies per second = 1 / Ping_Interval	1.91	1.91
	<b>Λ</b>	7.32E-09	9.05E-09
	<b>% of 1.00E-07 goal, up to 100% is OK</b>	7.32%	9.05%
	<b>PFH = Λ / 100</b>	7.32E-11	9.05E-11
	<b>% of SIL 3 (1.00E-07) PFD limit, up to 1% is OK</b>	0.0732%	0.0905%



## 2-8.1 Summary of PFH results for Extended

The following figure summarizes the worst-case result. The numbers are obtained from and the formulas above.

Figure 2-8.3 Summary of PFH Calculations

<b>Inputs</b>			
Expected Packet Interval (EPI)		0.010	Seconds
Ping Interval		0.524	Seconds
Number of Nodes (m)		63	
Multi-Cast Consumers		15	
Bit Error Rate (p)		0.001	
<b>Results</b>			
		<b>Single Cast</b>	<b>Multi Cast</b>
R(p)data <= 2 bytes	R(p) data section - 2 bytes of data	5.05E-22	5.05E-22
R(p)data >= 3 bytes	R(p) data section - 250 bytes of data	4.34E-25	4.34E-25
R(p)data	R(p) data section	5.05E-22	5.05E-22
R(p)ts	R(p)time stamp section	1.26E-22	1.26E-22
R(p)multi	R(p) Time correction message		1.31E-18
R(p)reply	R(p) Time coordination message	1.13E-18	1.13E-18
Msg_per sec	Messages per second = 1 / EPI	100	100
Rpl_per sec	Replies per second = 1 / Ping_Interval	1.00	1.00
	$\Lambda$	2.66E-11	8.18E-10
	% of 1.00E-07 goal, up to 100% is OK	0.0266%	0.8183%
	PFH = $\Lambda$ / 100	2.66E-13	8.18E-12
	% of SIL 3 (1.00E-07) PFD limit, up to 1% is OK	0.0003%	0.0082%

## 2-8.2 PFH Calculation for DeviceNet Safety Messages

Using the equations shown in section 2-8 it is possible to calculate the worst-case PFH. From the summary table presented in section 0 it can be seen that the 2 byte formats provide the worst case PFH for the system. Since this PDF/PFH is  $< 10^{-7}$  it clearly meets the SIL3 requirements for IEC 61508.



### 2-8.3 PFH Calculation for EtherNet/IP and SERCOS III Safety Messages

Using the equations shown in section 2-8 it is possible to calculate the worst case PFH. From the summary table presented in section 0 it can be seen that the 2 byte formats provide the worst case PFH for the system. Since this PDF/PFH is  $< 10^{-7}$  it clearly meets the SIL3 requirements for IEC 61508.

## 2-9 DeviceNet Requirements

### 2-9.1 Safety Network Implications on DeviceNet

This section provides the specific requirements to implement DeviceNet Safety.

#### 2-9.1.1 CAN Message Encoding

The safety message is encoded in a standard DeviceNet message. Figure 2-9.1 shows the format of the message.

**Figure 2-9.1 Standard CAN Message Encoding Showing Safety Message**

Physical Addressing							
Standard DeviceNet				Safety Message	Standard DeviceNet		
SOF	Identifier	RTR	Control		CRC	ACK	EOF
1 bit	11 bits	1 bit	6 bits		16 bits	2 bits	7 bits
Native CRC Coverage							

The following restrictions are placed on devices implementing DeviceNet Safety:

- Each DeviceNet Safety device shall always operate using a single MACID (node address).
- Only the safety configuration tool can set identifiers and parameters. If switches are used for these settings changes in the state of the switches are required to be monitored and verified against the expected configuration (refer to Chapter 9). The monitoring must be done during configuration, startup and run time.

#### 2-9.1.2 Use of DeviceNet Identifiers

SRS95 During connection processing of a SafetyOpen DeviceNet safety identifiers shall be assigned from the available pool of DeviceNet Group 1 identifiers .

The use of Group 1 identifiers ensures that safety messages have the highest priority during arbitration. The 12 lowest numbered DeviceNet identifiers are available for use by safety connections. The remaining 4 Group 1 identifiers are reserved for standard connections (from the existing predefined Master-Slave connection set). This partitioning ensures that all of the safety messages will be of a higher priority during arbitration.

Table 2-9.1 shows how the safety connections use identifiers with MSGID in the range of 0 to B (hex).



**Table 2-9.1 Group1 identifier space**

	Group1 identifiers		
	bit 10 =0	bits 9,8,7,6 MSGID	bits 5,4,3,2,1,0 MACID
Group1 ID used for safety	0	0000	Src OR Dest MACID
Group1 ID used for safety	0	0001	Src OR Dest MACID
Group1 ID used for safety	0	0010	Src OR Dest MACID
Group1 ID used for safety	0	0011	Src OR Dest MACID
Group1 ID used for safety	0	0100	Src OR Dest MACID
Group1 ID used for safety	0	0101	Src OR Dest MACID
Group1 ID used for safety	0	0110	Src OR Dest MACID
Group1 ID used for safety	0	0111	Src OR Dest MACID
Group1 ID used for safety	0	1000	Src OR Dest MACID
Group1 ID used for safety	0	1001	Src OR Dest MACID
Group1 ID used for safety	0	1010	Src OR Dest MACID
Group1 ID used for safety	0	1011	Src OR Dest MACID
Slave's I/O Multi-Cast poll response message	0	1100	Src MACID
Slave's I/O COS or cyclic message	0	1101	Src MACID
Slave's I/O Bit strobe response	0	1110	Src MACID
Slave's I/O Poll response or COS/cyclic acknowledge message	0	1111	Src MACID

In order to preserve connection ID's, the rules in the following sections shall be implemented for the allocation of connection ID's.

Typically the connection originator is a client device like a PLC or Safety Monitor, while the target is a server device like an I/O module. The client device will typically maintain more connections than the server device(s). This new (DeviceNet specific) algorithm will help achieve more total network connections than the standard CIP allocation model.

#### 2-9.1.2.1 Basic DeviceNet Algorithm (General Model)

SRS96 Safety connection originators shall initially ask the target (via SafetyOpen) to allocate one, two, or three identifiers according to the rules stated in Table 24.

(This is accomplished by inserting 0xFFFFFFFF identifiers in the SafetyOpen request sent to the target)

SRS97 If a safety target cannot or refuses to allocate the identifier(s), the target shall respond to the Safety Open request with a 0x01 general error status with extended status 0x031F indicating no connection resources are available.

The originator receives the SafetyOpen response.

SRS98 If the target has allocated identifiers from its pool, it shall insert them into the SafetyOpen success response.

The originator accepts these and the connection is completed.



If the target has not allocated identifiers from its pool then the originator has the option to allocate identifiers from its own pool. If the originator does allocate the identifier(s) then it must convey this data to the target (via a 2<sup>nd</sup> Safety Open)

If the originator cannot or refuses to allocate an identifier then the connection cannot be established.

#### **2-9.1.2.2 Case 1 (Target Allocates Identifiers)**

This first scenario has the originator requesting that the target allocate the connection identifiers and the target agrees to do so.

The connection originator asks the target (via Safety Open) to allocate one, two or three identifiers.

- The target agrees to allocate the identifier(s) and responds to the Safety Open request with success.
- The originator receives the Safety Open response and accepts these identifiers and the connection is completed.

#### **2-9.1.2.3 Case 2 (Target Cannot Allocate Identifiers)**

This second scenario has the originator requesting that the target allocate the connection identifiers but the target cannot or refuses to do so.

- The connection originator asks the target (via Safety Open) to allocate one, two or three identifiers (identifier = 0xFFFFFFFF).
- The target chooses not to allocate the identifier(s) and responds to the Safety Open request with a 0x01 general error code with extended status 0x05 indicating that the resources are not available.
- The originator receives the Safety Open response, sees that target cannot allocate identifiers and then chooses to allocate identifiers from its own pool.
- The originator then constructs a 2<sup>nd</sup> Safety Open with the identifiers specified.
- The target accepts this Safety Open and the connection is completed.

#### **2-9.1.2.4 Case 3 (Originator Allocates Identifiers)**

This third scenario has the originator deciding to allocate its own identifiers with asking the target.

- The connection originator allocates the identifiers from its pool and enters these values into the Safety Open which is then sent to the target.
- The target accepts this Safety Open and responds with success.
- The originator receives the Safety Open response and the connection is completed.

#### **2-9.1.2.5 Order of MSGID Allocation**

As stated before each node has a total of 12 Group1 identifiers that can be used for safety connections.



SRS99 In order to provide consistency among producers and consumers, the following rule shall be used when allocating identifiers: The first MSGID to be allocated shall start at the highest available value and move downward.

Thus the first MSGID would be B(hex) or 1011(binary) and the next would be A(hex) or 1010(binary).

#### 2-9.1.2.6 Rules for Connection ID Assignment

The following rules for DeviceNet ID assignment shall be followed by Safety targets, originators and bridges. These rules provide repeatable ID allocation for a system where one or no routers are connected to DeviceNet. It effectively takes unused IDs from target devices and provides them to bridges. Because one resource, the router, is allowed to ask for these resources, it should work repeatably. But any attempt to extend this to other originators could result in a situation where repeatable power up was not achievable.

*Rule 1:* All on-link multicast originators provide the 3<sup>rd</sup> group 1 ID when establishing a connection.

*Rule 2:* All off-link multicast connection originators (routers) request the 3<sup>rd</sup> group 1 ID from the target

*Rule 3:* The group 1 ID in targets are divided into 2 types, *reserved* and *free*.

The *reserved IDs* are equal to 2 x the maximum number of safety connections that a device can support. The maximum number of connections is specified when a device is configured.

The number of *free IDs* is equal to Max group 1 ID (12) – *reserved IDs*.

In most target devices, this could be a default value and the user would never need to change it.

*Rule 4:* Each target would provide 2 reserved group 1 IDs to multicast or single cast originators until its *reserved* IDs are depleted. Once its reserved IDs are depleted it would reject requests to establish new connections. But it would still allow originators to try to establish multicast connection to multicast productions that were already established.

*Rule 5:* Each target would provide the 3<sup>rd</sup> group1 ID to multicast originators from its *free* pool. Once the *free* pool is exhausted, it would reject the request and the originator would need to supply the 3<sup>rd</sup> ID



Table 2-9.2 DeviceNet ID Assignment Rules

ConnectionType	Intermediate (First) Hop	Last (Only) Hop 1 <sup>st</sup> attempt	Last (Only) Hop 2 <sup>nd</sup> attempt	Last (Only) Hop 3 <sup>rd</sup> attempt
Singlecast	T ⇒ O: Target Router O ⇒ T: Originator or Originating Router	T ⇒ O: Target O ⇒ T: Target	Connection Fault, Connection cannot be established.	Connection Fault, Connection cannot be established.
Multi-cast T ⇒ O (typical; input module, output module status)	T ⇒ O: Target Router O ⇒ T: Originator or Originating Router TC: Target Router	T ⇒ O: Target O ⇒ T: Conditional** TC: Target	T ⇒ O: Target O ⇒ T: Originator (if first request was Target) TC: Target	Connection Fault, Connection cannot be established.
Multi-cast O ⇒ T	T ⇒ O: Target Router O ⇒ T: Originator or Originating Router TC: Originator or Originating Router	T ⇒ O: Target O ⇒ T: Originator TC: Originator or Originating Router	T ⇒ O: Originator O ⇒ T: Originator TC: Originator or Originating Router	Connection Fault, Connection cannot be established

TC refers to the Time Correction Connection

\*\* Originators will default to providing this ID. If the CCO object is used, the originator will either provide this ID on the first request or ask the target for the ID based on the "ID Allocation" attribute in the CCO object (default is to provide the ID). Bridges will always ask the Target for this ID first.

Table 2-9.2 illustrates the rules for Connection ID Assignment. For non-bridged connections, the last 3 columns are of most interest. For example, an originator requesting a single cast connection, the target would be asked to supply both the T=>O and the O=>T connection IDs.

For a typical Multi-cast Input module, the connections are originated by the multi-cast consumer (i.e. logic device). In this case, the data is flowing from T=>O. Table 2-9.2 shows that the target (i.e. input module) is asked to allocate the IDs for 2 of connections when the originator is on-link, but could be asked to supply all 3 if the originator is off-link and routing the request through a bridge. This is done by the originator inserting 0xFFFF in the ID fields.

The SafetyOpen success response will support up to 3 IDs to be allocated by the target. If a target does not have available all the IDs being requested, the target should respond with Error Code 0x01, extended code 0x05 (Identifier resources not available). In this multi-cast case, the target also needs to recognize when the connection being requested already exists (i.e. multi-cast production) and may need to allocate identifiers already assigned.

Table 2-9.2 illustrates the rules for Connection Id Assignment. For non-bridged connections, the last 3 columns are of most interest. For example, an originator requesting a single cast connection, the target would be asked to supply both the T=>O and the O=>T connection Ids.

For a typical Multi-cast Input module, the connections are originated by the multi-cast consumer (i.e. logic device). In this case, the data is flowing from T=>O. Table 2-9.2 shows that the target (i.e. input module) is asked to allocate the Ids for 2 of connections when the originator is on-link, but could be asked to supply all 3 if the originator is off-link and routing the request through a bridge. This is done by the originator inserting 0xFFFF in the Id fields.



The SafetyOpen success response will support up to 3 Ids to be allocated by the target. If a target does not have available all the Ids being requested, the target should respond with Error Code 0x01, extended code 0x05 (Identifier resources not available). In this multi-cast case, the target also needs to recognize when the connection being requested already exists (i.e. multi-cast production) and may need to allocate identifiers already assigned.

The originator has the option of re-initiating the SafetyOpen and requesting less IDs. As illustrated in the table, originators who are multi-cast consumers can only allocate the ID for the O=>T identifier.

#### **2-9.1.2.7 Single-Cast Message Connections**

The basic algorithm described above applies to these two connection types.

#### **2-9.1.2.8 Multi-Cast Message Connections**

SRS100 Multi-Cast producers must allocate an identifier from its own pool, i.e. the source MACID must be that of the producer. Multi-cast producers shall reject any connection request that attempts to assign IDs for its produced data or time correction data. The multi-cast consumer may follow the basic algorithm described above.

#### **2-9.1.2.9 Why must the producer allocate the producing identifier?**

Assume the consumer allocated the identifier and the multi-cast producer begins producing on this identifier.

Then a second multi-cast consumer is added to the producer list.

Then the original consumer is shut down and restarted.

The multi-cast producer will detect the loss of the first connection but must keep producing data for the other connection.

Then when the original consumer is back on-line, the previously allocation identifier is unallocated (to the consumer viewpoint).

If another connection is made to this consumer, then this identifier could be reused and a network conflict would occur

#### **2-9.1.2.10 DeviceNet ID Quarantining Requirements**

Target devices need to take care that they don't re-allocate any group 1 IDs until they are certain that the IDs are no longer being used by the device they had been previously allocated to. To assure that this condition will not occur, the following algorithm must be followed:

SRS179 When a safety device closes a connection in which group 1 IDs were allocated to another device, the group 1 ID shall be placed into quarantine for a period equal to the (connection\_EPI \* Ping\_Interval\_Multiplier) \* (Timeout\_Multiplier.PI+2). During this time, the ID cannot be used or allocated to another device.



If an originator who borrowed an ID detects the communication failure and re-issues a new connection request before these IDs are removed from quarantine, devices can define schemes that detect these requests and pull the IDs from quarantine early and return them to the originator. However, care must be taken to assure that this is done with integrity.

#### 2-9.1.2.11 Restrictions on Standard DeviceNet Features

Because of the potential for safety node disruption, Safety devices should not implement the DeviceNet Quick Connect feature.

### 2-9.2 Safety System Throughput Over DeviceNet

#### 2-9.2.1 DeviceNet Safety Performance

The performance of producer-consumer I/O connections on DeviceNet Safety depends on:

- The baud rate of the network,
- The packet size for the various connections,
- The type and number of safety connections used.

##### 2-9.2.1.1 Application Example 1

The sample application shown in Figure 2-9.2 will be used to estimate the network performance for an application and maximum EPI that can be supported for a given baud rate, and reserved bandwidth percentage. The reserved bandwidth is required to allow for standard traffic for connection establishment, diagnostics, and standard Class 3 messaging.

Figure 2-9.2 Sample Application



For example, a typical safety input might be configured to use a multi-cast or a uni-directional point-to-point safety connection to produce safety data to multiple consumers or one consumer respectively. Which is used will affect the maximum EPI of the application. A safety network software tool, interested in determining the number of safety nodes and reaction time that the application can support, can analyze the application information to provide the user with reaction times and best case EPI values.



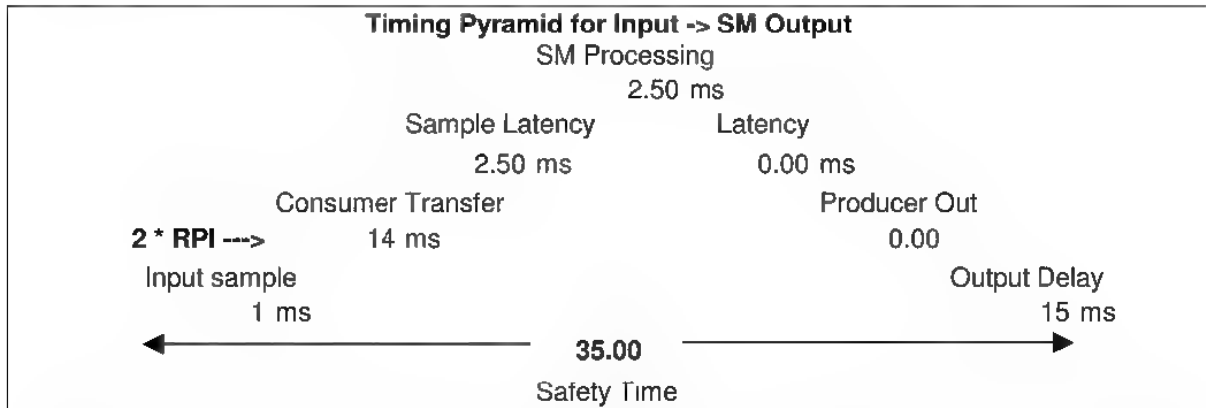
**Table 2-9.3 Estimated Application Performance**

		UniDir	Multicast	Bi-Dir	Total
Connections Required		16	0	0	
Baud Rate	500K				
Reserved Bandwidth	40%				
Safety Payload Size (in bytes)		1	1	1	
Bytes per production		7	7	26	
Total Bits per production		125	125	478	
Total Bits for all connections		2000	0	0	2000
Number of productions per second					150
Best EPI in msec, rounded up					7

In this example, the baud rate is assumed to be 500K, bit stuffing is assumed to be a worst-case 20%, and the reserved bandwidth is set to 40%. After all of the calculations are made, the best EPI would be approximately 7 mSec. The EPI value is rounded up to the nearest integer.

Once these numbers are known, they can be put into a Safety Loop Timing tree to calculate the reaction time of the application. Estimated scan times for the Safety Monitor (SM) is set at 2.5 ms.

**Figure 2-9.3 Example Safety Loop Timing Tree**



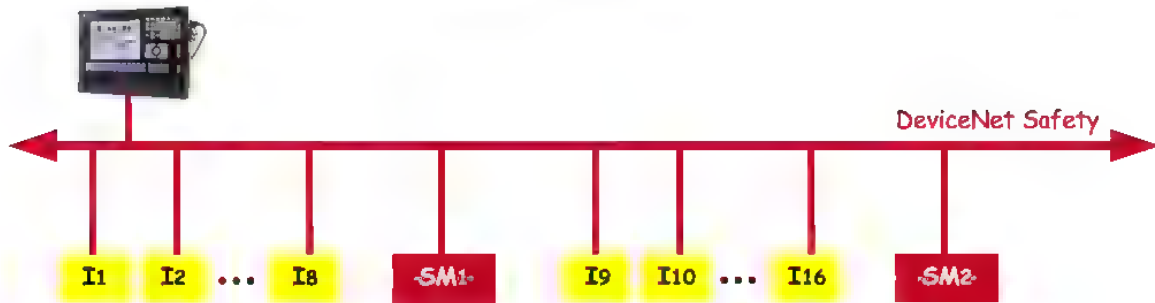
The safety loop time is worst case in that the network transfer times approximate worst-case watchdog time limits and that the logic engine is running asynchronous to the network updates.

In this application example, the standard DeviceNet messaging is neglected in these equations. The assumption is that the 40% reserved bandwidth will be sufficient for infrequent Class 3 DeviceNet. This assumption may not be true in cases where high bandwidth utilization is assumed or Master/Slave is also on the same wire. This application also assumes that ping responses are much slower than the safety EPI and would sent using the reserved bandwidth.



### 2-9.2.1.2 Application Example 2

Figure 2-9.4 Application with Interlocked Safety Monitors



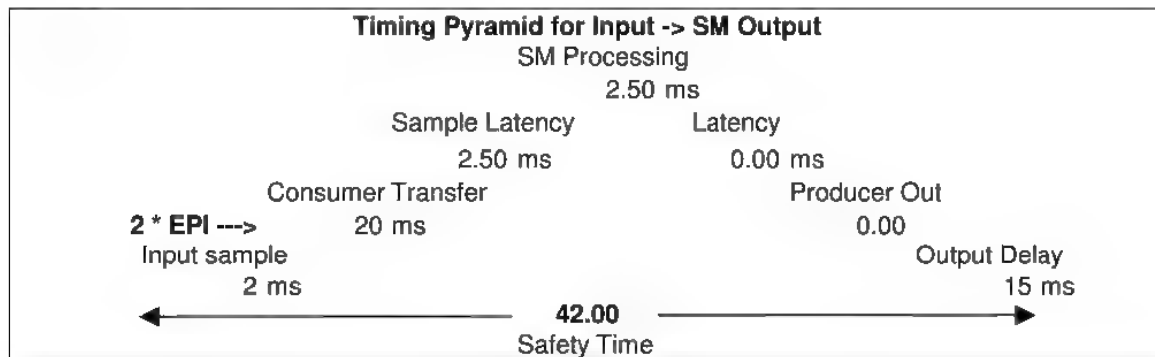
A second example application is a combined Safety and Standard DeviceNet application. The Safety application is 7 inputs to one Safety Monitor with local outputs. Two bytes of safety payload data on single-cast connections are assumed. On standard DeviceNet, the application has a single Scanner, 16 Bit-strobe Inputs and 16 Polled outputs. 2 bytes of standard data are transferred between each node. Using a reserved bandwidth of 80% (70% for standard, 20% Safety and 10% margin), we calculate the following Safety EPI:

Table 2-9.4 Estimated Application Performance

		UniDir	Multi-cast	Bi-Dir	Total
Connections Required		7	0	0	
Baud Rate	500K				
Reserved Bandwidth	80%				
Safe Payload Size (in bytes)		2	1	1	
Bytes per production		8	7	26	
Total Bits per production		134	125	478	
Total Bits for all connections		938	0	0	938
Number of productions per second					106
Best EPI in msec, rounded up					10

The reaction time would be:

Figure 2-9.5 Timing Pyramid for Interlocked Safety Monitors

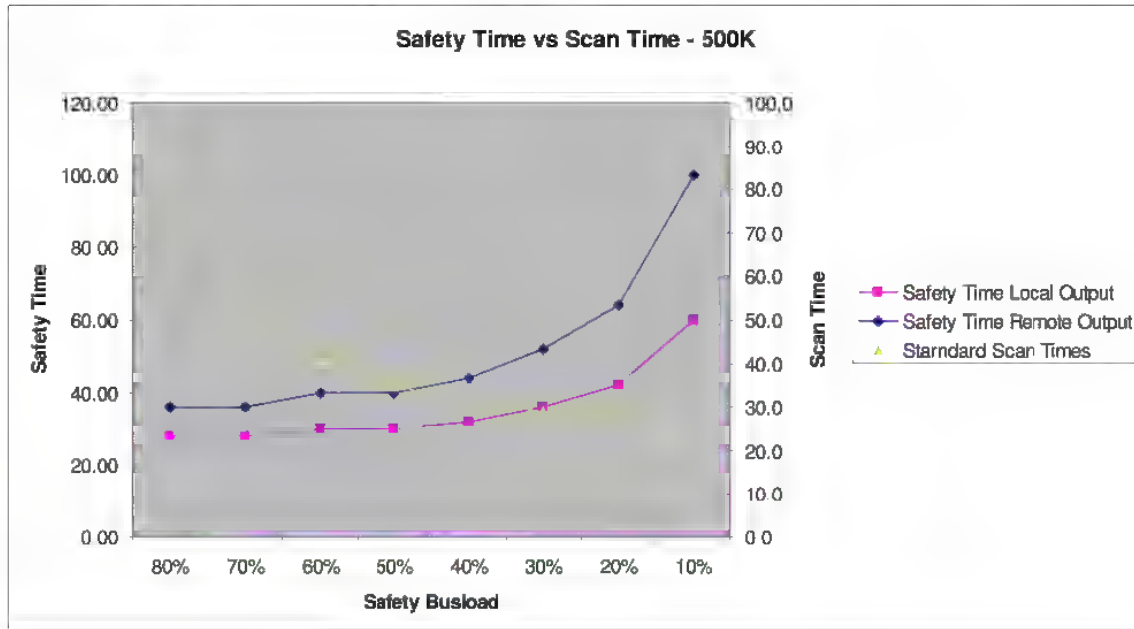




If we were to do this calculation over a number of bandwidth allocations and plot the reaction time vs. Standard Scanner Scan Time, we would get a graph like that shown in Figure 2-9.6. Giving Safety 40 - 50% of the busload provides both the Scanner and Safety a reasonable level of performance.

NOTE: When Safety and Standard DeviceNet Scanning are sharing the same subnet, Safety EPI calculations should never allocate more than 50% of the busload. This assures that all Safety nodes will have sufficient opportunity to send each EPI under worse case conditions.

Figure 2-9.6 reaction time vs. Scan Time



### 2-9.3 Bus Structure

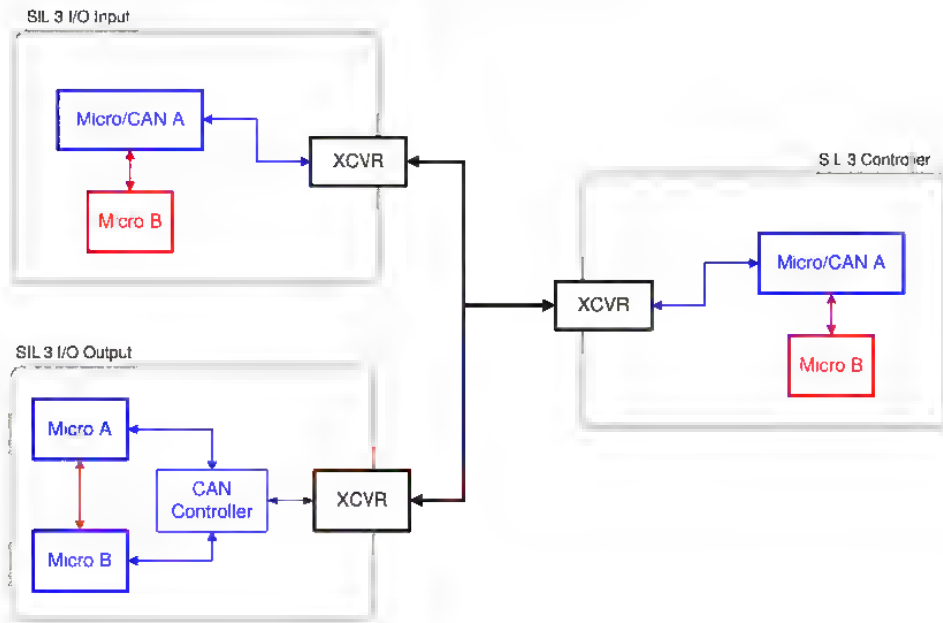
An example system is shown in Figure 2-9.7. A variety of architectures are possible. It shows devices with 1 CAN interface implementing a single CAN bus structure. This bus structure is made up of 2 CPUs, 1 CAN interface chips and 1 CAN transceiver. Another example is shown with 2 CPUs with 1 CAN controllers and 1 CAN transceiver.

Each CAN interface shall have a unique message ID for each safety connection.

End devices can support more than one safety connection. This may be desirable since some errors will cause the connection to shut down. In the case of multiple safety connections, each connection shall support its own safety configuration. Thus, each connection may have a different owner and periodic rate.



Figure 2-9.7 Example of DeviceNet interconnection





## **2-10 EtherNet/IP Requirements**

This section contains requirements and advisory information for Safety devices that reside directly on an EtherNet/IP network.

### **2-10.1 EPI Rules for Safety Messages That Travel Over EtherNet/IP**

Safety packets that must travel over EtherNet/IP have certain restrictions based on the Safety format used when establishing the connection.

Safety Connection using the base format and traveling over EtherNet/IP are restricted to EPIs less than 100ms. The method by which this restriction is enforced is a vendor specific decision. For example, the restriction could be enforced in the originator configuration software or in originator connection establishment checks. Vendors will be responsible for verifying that proper restrictions are in place.

### **2-10.2 Default Safety I/O Service**

Safety connections support both Point-to-point and Multicast connection types, but EtherNet/IP has a number of system complications when Multicast services are used. It is highly recommended that when configuring safety connections in an originator, the safety configuration software should present point-to-point as the default service for all Safety configurations. The User should be given the option to select Multicast, but if this selection is not specifically made, point-to-point should be used.

Safety EtherNet/IP devices should consider adding support for multiple point-to-point connections to the same safety input assembly to better facilitate the use of point-to-point.

### **2-10.3 Duplicate IP Detection**

Safety devices that reside on EtherNet/IP shall support the IPv4 Address Conflict Detection mechanism defined in ODVA PUB00127.

### **2-10.4 Priority for Safety Connections**

EtherNet/IP Safety devices must utilize the standard EtherNet/IP Quality of Service scheme; see Volume 2, Chapters 3 and 5 for implementation details. EtherNet/IP Safety Devices must implement the QoS Object and support priority marking via DiffServ Code Points (DSCP). EtherNet/IP Safety devices shall use “scheduled” as the default priority for safety I/O devices.

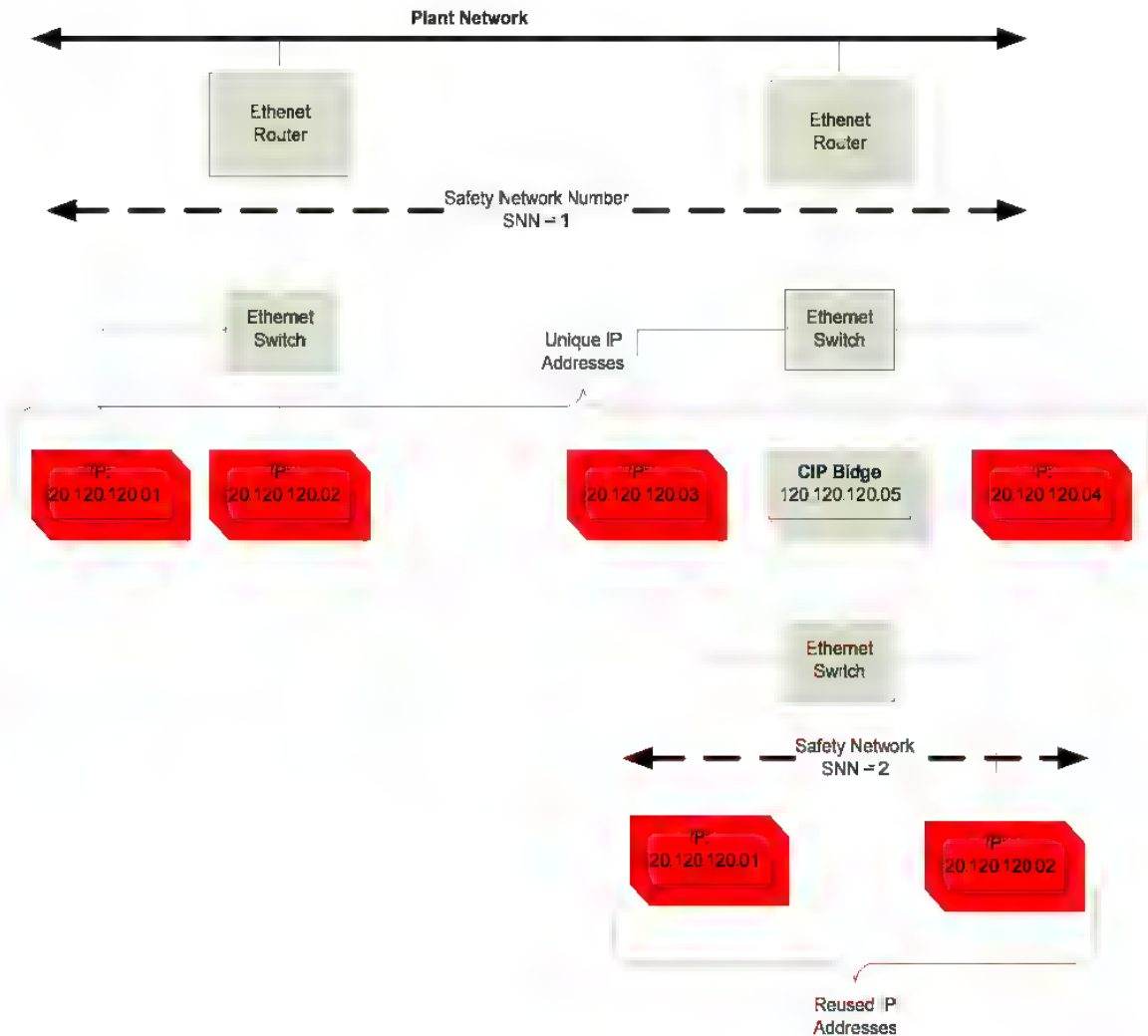
### **2-10.5 EtherNet/IP Networks and Safety Network Numbers**

Safety requires the assignment of Safety Network Numbers (SNN) in all Safety EtherNet/IP nodes. The SNN is combined with the nodes’s IP address to form a plantwide Unique Node ID (UNID). An example showing when a new SNN must be assigned is shown in Figure 2-10.1.



Figure 2-10.1 shows that nodes separated by switches and/or routers are still on the same network because they all have unique IP addresses. These nodes can all use the same SNN. However, when a subnet is created (i.e through a CIP bridge) so that IP addresses can be reused, a new SNN must be assigned. *The indicator that a new SNN must be assigned is the re-use of IP addresses on the network.*

**Figure 2-10.1 EtherNet/IP Networks and SNN Assignment**





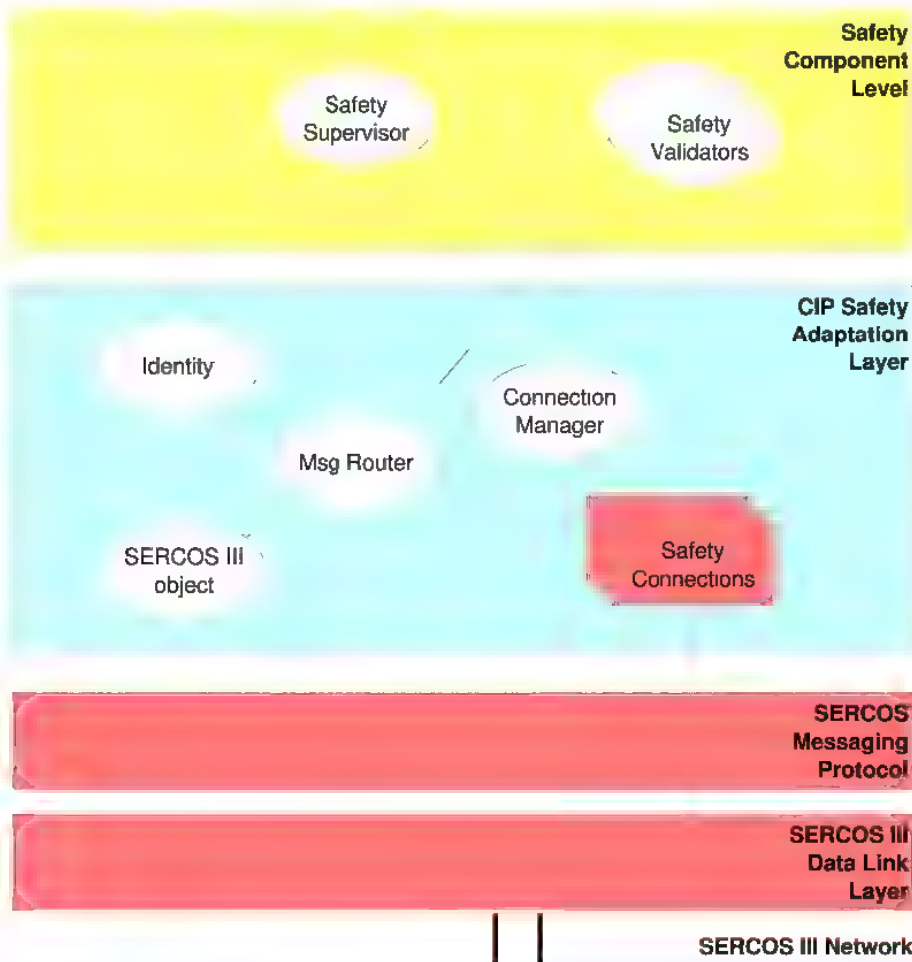
## 2-11 SERCOS Requirements

This section contains requirements and advisory information for Safety devices that reside directly on a SERCOS network and modular SERCOS Devices that contain safety modules.

### 2-11.1 Baseline CIP Safety on SERCOS Device

The CIP functionality needed to implement CIP Safety in a SERCOS III device is encapsulated in an adaptation layer on top of the standard SERCOS III communication mechanisms. The SERCOS III function group “S 0 1830 Safety Connection” provides the functionality for configuring both explicit and I/O connections, and for exchanging data via the SERCOS Messaging Protocol (SMP).

Figure 2-11.1 Baseline CIP Safety on SERCOS device



The CIP Safety Adaptation Layer provides CIP functionality similar to the Baseline Safety Device defined in Chapter 6-3.



## 2-11.2 Transport Layer Requirements

CIP Safety on SERCOS devices shall use the SERCOS Messaging Protocol (SMP) to exchange safety messages. SMP defines a set of messaging containers that can be transported via SERCOS III connections. A detailed description of the transport layer is presented in the SERCOS III specification (first published with V1.1.2) by SERCOS International (SI).

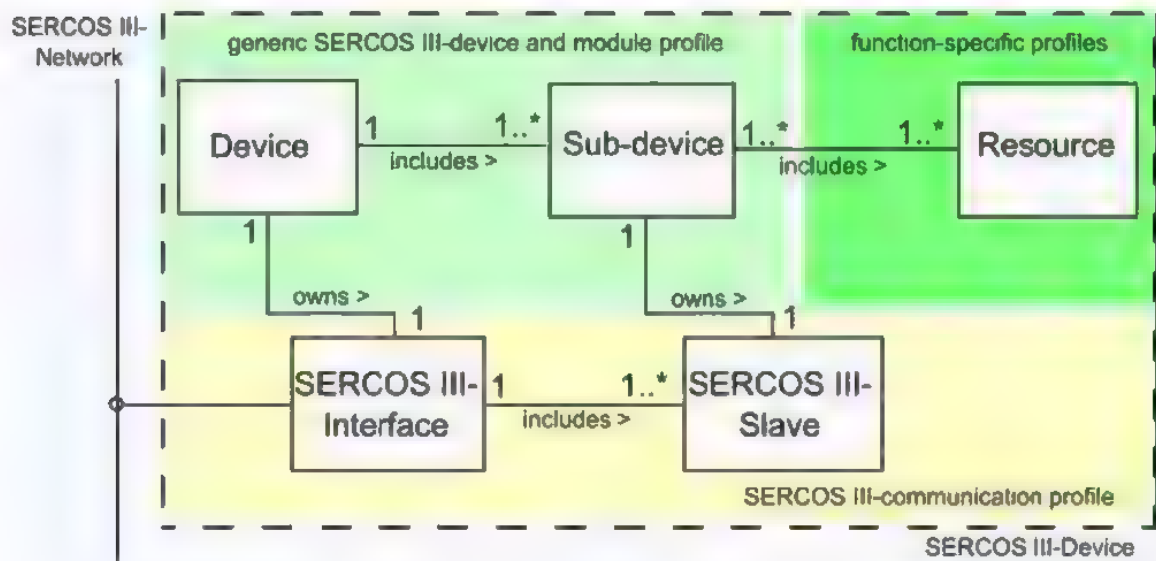
## 2-11.3 Multicast Connections

SERCOS III connections may be used to build point-to-point or multi-cast safety connections.

## 2-11.4 CIP Safety and the SERCOS III device model

Figure 2-11.2 shows the components of a generic SERCOS III device. There is a clear separation into communication- and device-related components. The communication-related components are defined in the SERCOS III communication profile (SCP). The SERCOS III Interface component represents the physical interface to the SERCOS III network, while the SERCOS III Slave component provides a logical interface with a SERCOS Slave Address that is unique to the SERCOS III Network. This structure allows for a modular device design with several sub-devices sharing a single physical interface.

Figure 2-11.2 SERCOS III device model<sup>5</sup>



In this context, “modular design” does not imply that the device hardware is modular as well. A monolithic device may internally be regarded as being modular if it features several independent sub-devices. For example, a dual axis AC drive controllers may internally be modeled as a single device containing two independent sub-devices with two different SERCOS addresses that are connected to the SERCOS III network via the same interface.

<sup>5</sup> The multiplicity of instances in this diagram is shown conforming to the UML 2.0 syntax. “1..\*” means that one or more instances of this entity are participating in this association.



In SERCOS III terms, safety functionality is modeled as a function-specific profile that implements one or more resources in a sub-device. This implies that a SERCOS III Device may contain several independent Safety Devices as defined by the CIP Safety specification (refer to the entity relationship diagrams in Volume 5, Chapter 2-1 and 2-2). As there is a substantial difference between a “Device” and a “Safety Device” in the SERCOS III context, care should be taken not to mix up these terms.

Modular I/O devices are modeled as a single SERCOS III Device with usually one single sub-device. Each I/O module is represented by a set of function groups in this sub-device.

## **2-11.5 UNID Assignment on SERCOS III**

CIP Safety requires the assignment of a plantwide Unique Node ID (UNID) to each safety device. The 10 byte UNID consists of a 6 byte Safety Network Number (SNN) combined with the node’s 4 byte network address (MACID on DeviceNet, IP address on EtherNet/IP).

For a SERCOS III Safety Device, the MACID/IP address portion of the UNID shall be replaced by a 4 byte SERCOS III Safety Device ID (SDID). Both the SDID and the SNN shall be stored as attributes of the SERCOS III object.

```
struct UNID
{
    S_SNN    SNN;
    UDINT    SDID;
}
```

The following rules shall be applied when assigning the UNID to a SERCOS III Safety Device.

### **2-11.5.1 SERCOS III Safety Device ID**

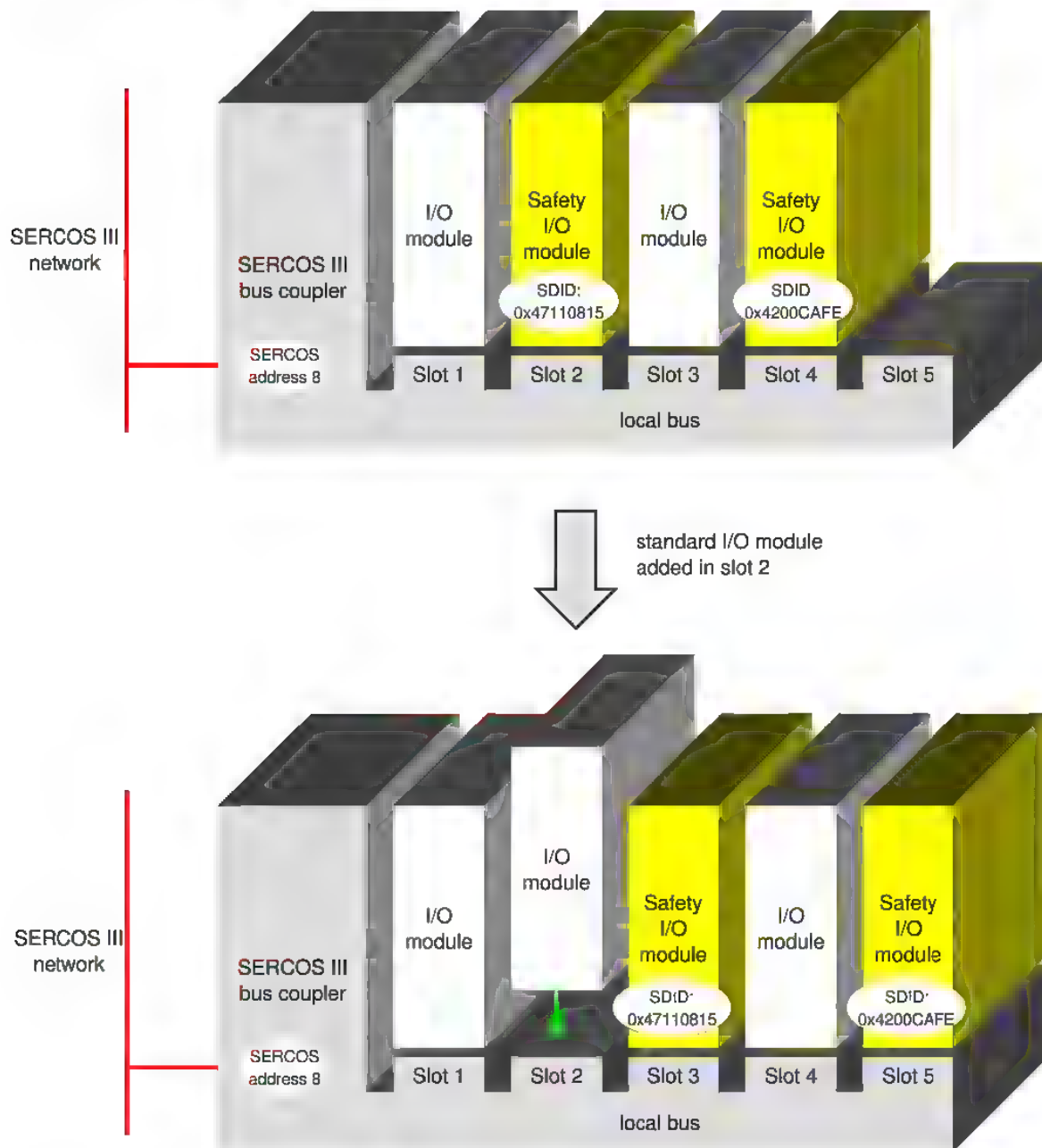
The preceding section on the SERCOS III device model showed that it is impossible to uniquely identify a SERCOS III safety device in a modular SERCOS device by its network node address (SERCOS address, SERCO parameter S-0-1040).

In addition, excluding the network node address from the UNID leads to a better separation of the standard device configuration from the safety configuration. This facilitates device commissioning, because a module may be moved to a different physical location after the safety configuration has been verified and locked. Only the non-safety-relevant parts of the configuration have to be adjusted to the new network topology by changing the SERCOS addresses. The safety layer will still ensure that only safety connections configured for this safety device can be established.

Therefore, in a SERCOS III network, the SERCOS III Safety Device ID portion of the UNID shall be chosen by the configuration tool or set in the safety device by other means (e.g. DIP switches), and shall not be related to neither the SERCOS node address (e.g. node addr 8 in Figure 2-11.3), nor the physical position of a module on the device’s local bus (e.g. slot number in Figure 2-11.3).



Figure 2-11.3 Adding a standard module to a modular device



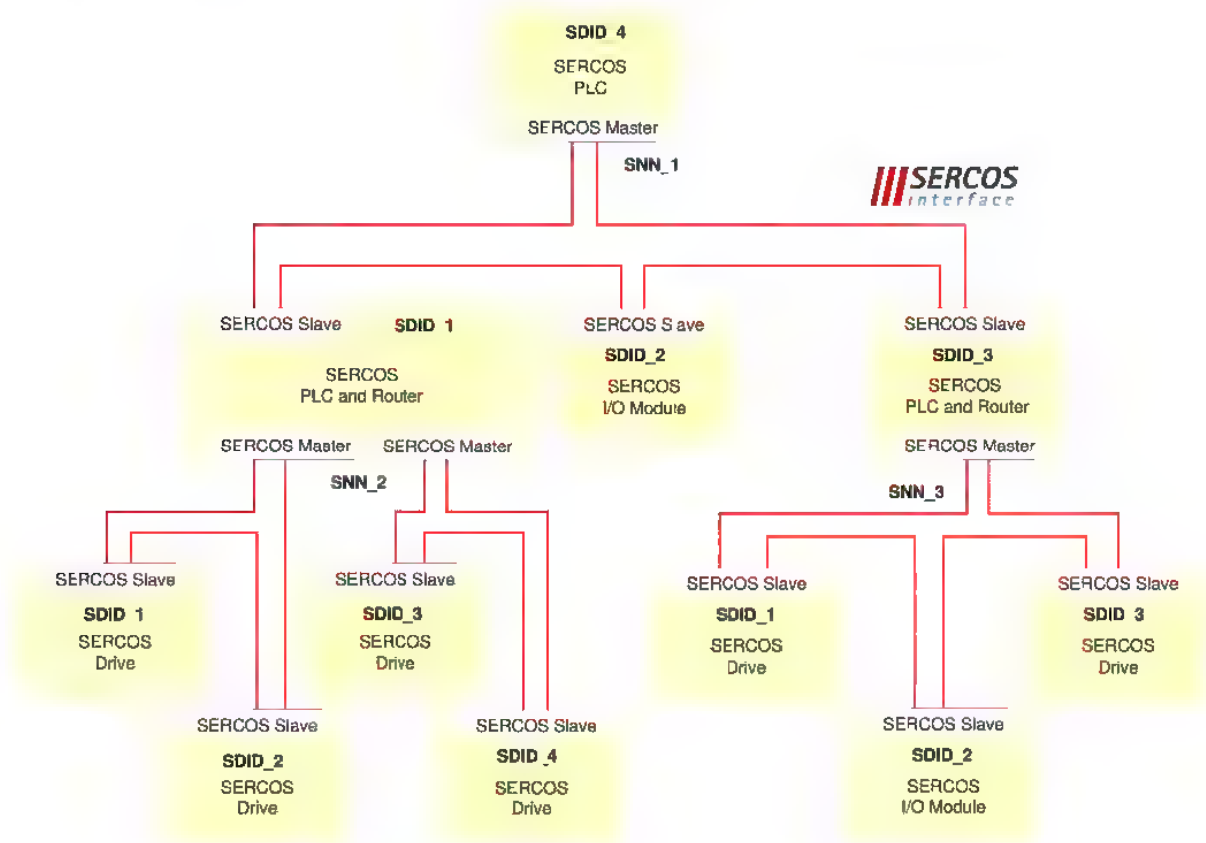
It has to be assured that the cabling attached to a module is also moved to the new location when moving a safety module without updating the safety network configuration. The manufacturer of a CIP Safety on SERCOS device shall ensure this by appropriate means or by adding an appropriate advisory to the device's safety manual.



### 2-11.5.2 Safety Network Number Assignment

CIP Safety requires the assignment of Safety Network Numbers (SNN) in all CIP Safety on SERCOS nodes. SNN and SDID together form the UNID that has to be unique for each device in a safety system. Because the SDID portion of the UNID is not related to the standard network address of a safety device, a SNN value is not limited to the boundaries of single SERCOS III network. Several SERCOS III networks may use the same SNN as long as they belong to the same configuration segment (a group of SERCOS III networks configured by a single configuration tool). The configuration tool shall be responsible for assigning different SDID values to all safety devices in one configuration segment.

Figure 2-11.4 Assigning Safety Network Numbers





## **Volume 5: CIP Safety**

# **Chapter 3: Communications Objects**



## Contents

3-1	Introduction.....	3
3-2	Connection Object .....	3
3-2.1	Class Attribute Extensions .....	3
3-2.2	Service Extensions .....	4
3-2.2.1	Explicit Message Response Format for SafetyOpen & SafetyClose .....	4
3-3	Connection Manager Object .....	5
3-3.1	Forward Open for Safety.....	5
3-3.2	SafetyOpen for EF Format .....	7
3-3.3	Originator Rules for Calculating the Connection Parameter CRC .....	7
3-3.4	SafetyOpen Processing Flowcharts .....	7
3-3.5	Checks required by Multi-cast Producers with existing connections .....	11
3-3.6	Electronic Key Usage for Safety .....	11
3-3.7	RPI vs. API in Safety Connections .....	11
3-3.8	Application Path Construction for Safety .....	12
3-3.9	Safety Validator Connection Types .....	13
3-3.10	Application Reply Data in a Successful SafetyOpen Response .....	15
3-3.11	Unsuccessful SafetyOpen Response for Safety .....	17
3-3.11.1	Safety-Specific Error Responses .....	17
3-3.12	Forward Close for Safety .....	19



## 3-1 Introduction

This chapter defines the extensions to the communication objects for CIP Safety. Safety connections use Class 0 connections for I/O messages. Standard Explicit message connections are used for the establishment of Safety I/O connections.

## 3-2 Connection Object

Several safety specific class attributes and services are defined to support CIP Safety. DeviceNet Safety devices use these components for connection management.

SRS57 Safety Nodes which reside on DeviceNet shall implement the SafetyOpen and SafetyClose services of the Connection Object.

### 3-2.1 Class Attribute Extensions

The following class attributes are defined for use by CIP Safety devices.

Attrib ID	Need in implementation	Access Rule	NV	Name	Date Type	Description of Attribute
8	Optional	Get	V	Connection request error count	UINT	Diagnostic Counter that is a running count of SafetyOpen or SafetyClose Faults.
9	Optional	Get	V	Safety Connection counters	STRUCT of	Status counters
				SafetyOpen Request success count	UINT	Count of successful SafetyOpen requests issued from this node to another target.
				SafetyOpen Request failure count	UINT	Count of unsuccessful SafetyOpen requests issued from this node to another target.
				SafetyClose Request success count	UINT	Count of successful SafetyClose requests issued from this node to another target.
				SafetyClose Request failure count <sup>2</sup>	UINT	Count of unsuccessful SafetyClose requests issued from this node to another target
				SafetyOpen Indication success count	UINT	Count of successful SafetyOpen requests issued from an external originator to this node
				SafetyOpen Indication failure count <sup>2</sup>	UINT	Count of unsuccessful SafetyOpen requests issued from an external originator to this node.
				SafetyClose Indication success count	UINT	Count of successful SafetyClose requests issued from an external originator to this node
				SafetyClose Indication failure count <sup>2</sup>	UINT	Count of unsuccessful SafetyClose requests issued from an external originator to this node.

<sup>1</sup>Attribute 8 is the sum of the 4 failure counters in attribute 9. Having this single attribute represent connection errors can be used in monitoring software to efficiently monitor errors without polling the additional attribute 9 structure. Once attribute 8 is non-zero then the monitoring software can examine attribute 9 for more details

<sup>2</sup>Included in class attribute 8.



### 3-2.2 Service Extensions

The following services are defined for use by CIP Safety devices on DeviceNet. These services shall be supported by a target device on DeviceNet. The SafetyOpen and SafetyClose services, used in lieu of the Forward\_Open and Forward\_Close services of the Connection Manager object, inherit the identical behavior from those Connection Manager services.

**Table 3-2.1 Service Extensions**

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
4D <sub>hex</sub>	Optional	N/A	Reset Counters	This service will reset class attributes 8 and 9 to zero.
4E <sub>hex</sub>	Required	N/A	SafetyClose	Used to close a Safety I/O connection. See the Forward_Close service of the Connection Manager object.
54 <sub>hex</sub>	Required	N/A	SafetyOpen <sup>1</sup>	Used to establish a safety I/O connection. See the Forward_Open service of the Connection Manager object

<sup>1</sup> The SafetyOpen service data shall contain a Safety Network Segment in the Connection\_Path field.

#### 3-2.2.1 Explicit Message Response Format for SafetyOpen & SafetyClose

The SafetyOpen and SafetyClose service responses have a common response format. This format contains status information for both success and failure. If the service response is a success, response data specific to each service is returned. If the service response is an error, a successful DeviceNet status, the general error code returned is 0xFF, along with a variable amount of extended error information is returned.

The SafetyOpen and SafetyClose Object Specific services have the following general response format:

**Table 3-2.2 SafetyOpen & SafetyClose response format**

Parameter Name	Data Type	Description of Response Parameter
Reserved	USINT	Reserved
General Status	USINT	One of the General Status Codes listed in Appendix B (Status Codes).
Size of Additional Status	USINT	Number of 16 bit words in Additional Status array
Additional Status	ARRAY of UINT	Additional status
Response Data	ARRAY of octet	Response data from request or additional error data if General Status indicated an error.

The Response Data and General/Additional Statuses for the SafetyOpen and SafetyClose services shall match the Response Data and General/Additional Statuses for the Forward\_Open and Forward\_Close services of the Connection Manager object, respectively.



### 3-3 Connection Manager Object

Safety devices shall use a Forward\_Open with a Safety Network Segment to establish safety connections. This segment uniquely identifies the Forward\_Open service as a safety Forward\_Open. Throughout this section, the term “Safety Open” shall refer to both the Forward\_Open service of the Connection Manager object (when containing a Safety Network Segment) and the SafetyOpen service of the Connection object.

#### 3-3.1 Forward\_Open for Safety

The contents of the Forward\_Open service of the Connection Manager object, including the safety network segment, are outlined in Table 3-3.1.

**Table 3-3.1 Forward\_Open with Safety Network Segment**

Parameter Name		Data Type / Size	Target Destination & Semantics
Priority/Time_tick		BYTE	Connection Manager Servicing Request
Time-out_ticks		USINT	Connection Manager Servicing Request
O_to_T Network Connection ID		UDINT	Consuming Connection Instance : CIP Consumed Connection ID
T to O Network Connection ID		UDINT	Producing Connection Instance. CIP Producing Connection ID
Connection Serial Number		UINT	Value: Unique (to originator) 16-bit number Destination: Connection Manager Servicing Request
Originator Vendor ID		UINT	Connection Manager Servicing Request
Originator Serial Number		UDINT	Connection Manager Servicing Request
Connection Timeout Multiplier		USINT	Producing & Consuming Connection Instance Timeout Multiplier
Reserved		3 octets	
O_to_T RPI		UDINT	Consuming Connection Instance: EPR
O to T Network Connection Parameters	Redundant Owner	WORD	Value: 0 (Safety Connections do not support redundancy), Destination: Connection Manager
	Connection Type		Value: Point to Point; Destination: Connection Manager
	Priority		Value: High Priority; Destination: Connection Manager
	Fixed/Variable		Value: Fixed; Destination: Connection Manager
	Connection Size (in bytes)		Value: Size of safety PDU including the Mode Byte, Actual, Complement, and Time stamp sections Destination: Consuming Connection Instance, Consumed Connection Size attribute
T to O RPI		UDINT	Producing Connection Instance: EPR
T to O Network Connection Parameters	Redundant Owner	WORD	Value: 0 (Safety Connections do not support redundancy); Destination: Connection Manager
	Connection Type		Value: Point to Point; Destination: Connection Manager
	Priority		Value: High Priority, Destination: Connection Manager
	Fixed/Variable		Value: Fixed; Destination: Connection Manager



Parameter Name		Data Type / Size	Target Destination & Semantics
	Connection Size (in bytes)		Value: Size of safety PDU including the Mode Byte, Actual, Complement, and Time stamp sections Destination: Producing Connection Instance; Produced Connection Size
Transport Type/Trigger		BYTE	Value: Safety single or multi cast Destination: Validator instance Trigger always set to Application Client/Server depending on direction
Connection_Path_Size		USINT	Number of 16 bit words in the Connection_Path field – destination is Connection Manager servicing request. Note: Value included in CRC calculation does not include the words used for Network/Port segments
Connection_Path (Padded EPATH)	Path	Variable	Consumed by intermediate bridge connection manager(s) Each Router Format-safety Network Segment immediately follows the (?-to-DeviceNet) bridge for which it is intended.
	Logical Segment: Electronic Key	Vendor ID	Value: Electronic Key of required target Servicing Connection Manager or Connection Object
		Prod Type	
		Prod Code	
		Compatible/ Major Rev	
		Minor Rev	
	Application Path 1	Logical Segment: Class	Refer to Section 3-3.8 for application path options Refer to CIP Specification, Volume 1, Chapter 3, Section 3-5.5.1.9 for rules on setting/parsing these paths
		Logical Segment: Instance	
	Application Path 2	Logical Segment: Connection Point	Refer to Section 3-3.8 for application path options Refer to CIP Specification, Volume 1, Chapter 3, Section 3-5.5.1.9 for rules on setting/parsing these paths
	Application Path 3	Logical Segment: Connection Point	Refer to Section 3-3.8 for application path options Refer to CIP Specification, Volume 1, Chapter 3, Section 3-5.5.1.9 for rules on setting/parsing these paths
	Config Data	Data Segment: Simple Data	Configuration data to apply to the Safety Application object identified by Config Path.
	Network Segment: Safety	Safety Segment Name	Safety Network Segment
		Segment Size	Target Format Size = 54 octets
		Segment Format	Target Format – 0
		Network Safety Data	Safety Network Segment data





### 3-3.2 SafetyOpen for EF Format

Safety Originators that support the Extended (EF) format may connect to Targets with RPI's of 100 ms or less with either the EF format or the existing format.

FRS361 Safety Originators that connect with an EtherNet/IP target or a DeviceNet target with an Ethernet/IP hop within the path and have an RPI of greater than 100 ms shall use the EF format.

A variety of methods may be used to assure this. EDS files and configuration software can pre-configure the choice, or the originator can use a dynamic algorithm to make the determinations. See Section 5-6.2.6 Connection Configuration Object, Format Type Attribute for more information.

The mechanism used by the originator to indicate to a target that the EF format is being requested will be with a new Safety Network Segment Format as defined in Appendix C.

### 3-3.3 Originator Rules for Calculating the Connection Parameter CRC

The Connection Parameter CRC (CPCRC) is only calculated when a connection is initially configured or when that connection configuration is changed (refer to the Connection Parameter CRC attribute in the Safety Supervisor object definition). This value remains static in the originator otherwise.

The connection path size in the SafetyOpen changes value depending on whether an originator is sending configuration data or not. Since this field is included in the CPCRC calculation, originators need to calculate and maintain two values when they support the SafetyOpen configuration function. One value is used when configuring a device while opening the connection and the other value is used when just opening the connection (no configuration data is included).

Also, the path size must be "pre-adjusted" to remove the path entries that get stripped off by routers as the Safety Forward\_Open traverses each sub-net. The requirement is that the originator is responsible for calculating the CPCRC value based on what the SafetyOpen or Forward\_Open will be when the target receives it so that the target can perform a simple calculation in all cases.

### 3-3.4 SafetyOpen Processing Flowcharts

The following flowcharts outline the required behavior of a target upon receipt of a SafetyOpen:

- A. Verify the Connection Parameter CRC. As depicted in Table 3-3.1, the connection parameter CRC covers the SafetyOpen fields that will be used by the target. Routers along the path shall not modify these fields.
- B. The target checks that the electronic key segment matches the target. See 3-3.6.
- C. Depending on the type of safety connection, the logic is shown in Figure 3-3.1 or Figure 3-3.2 shall be followed.



Figure 3-3.1 Target Processing SafetyOpen with no configuration data (Type 2 SafetyOpen).

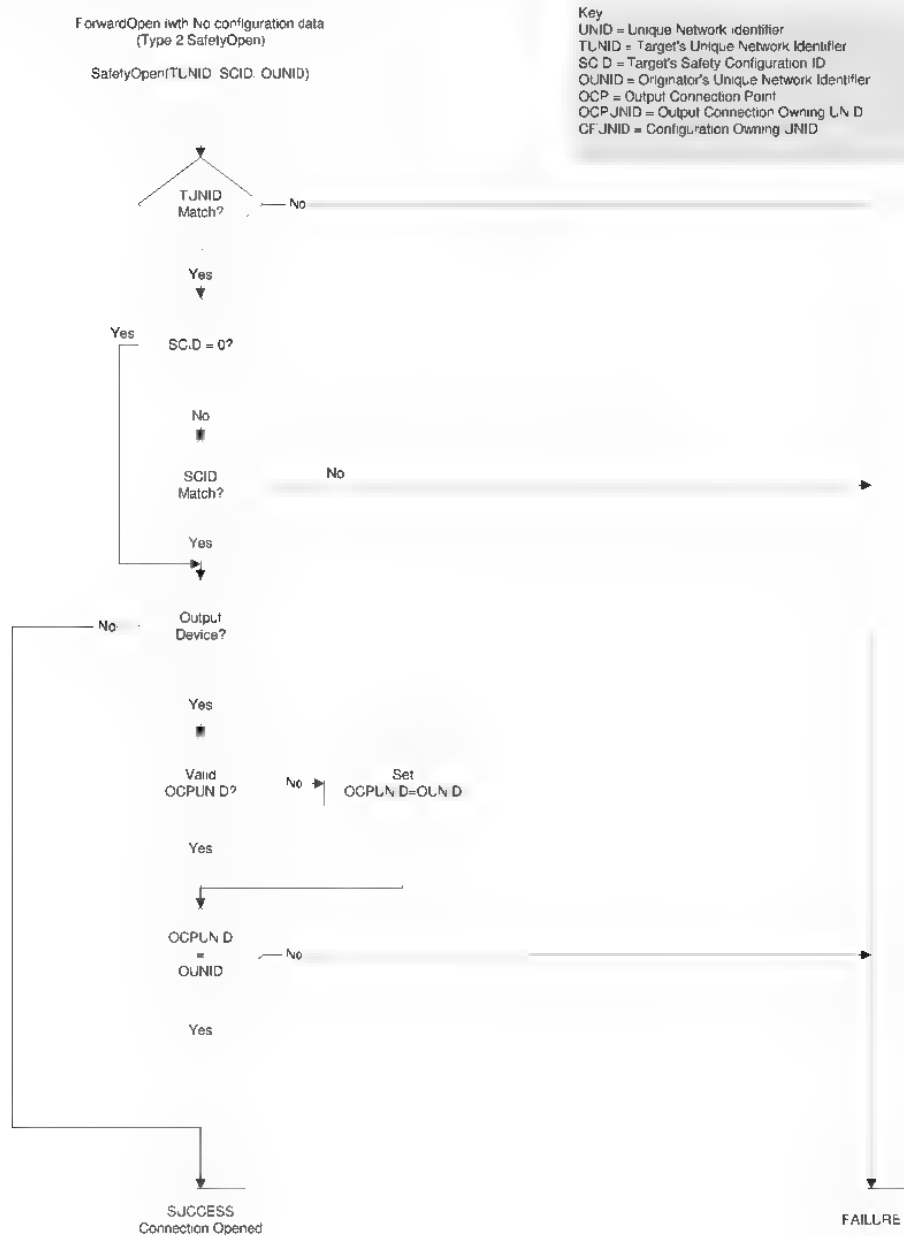
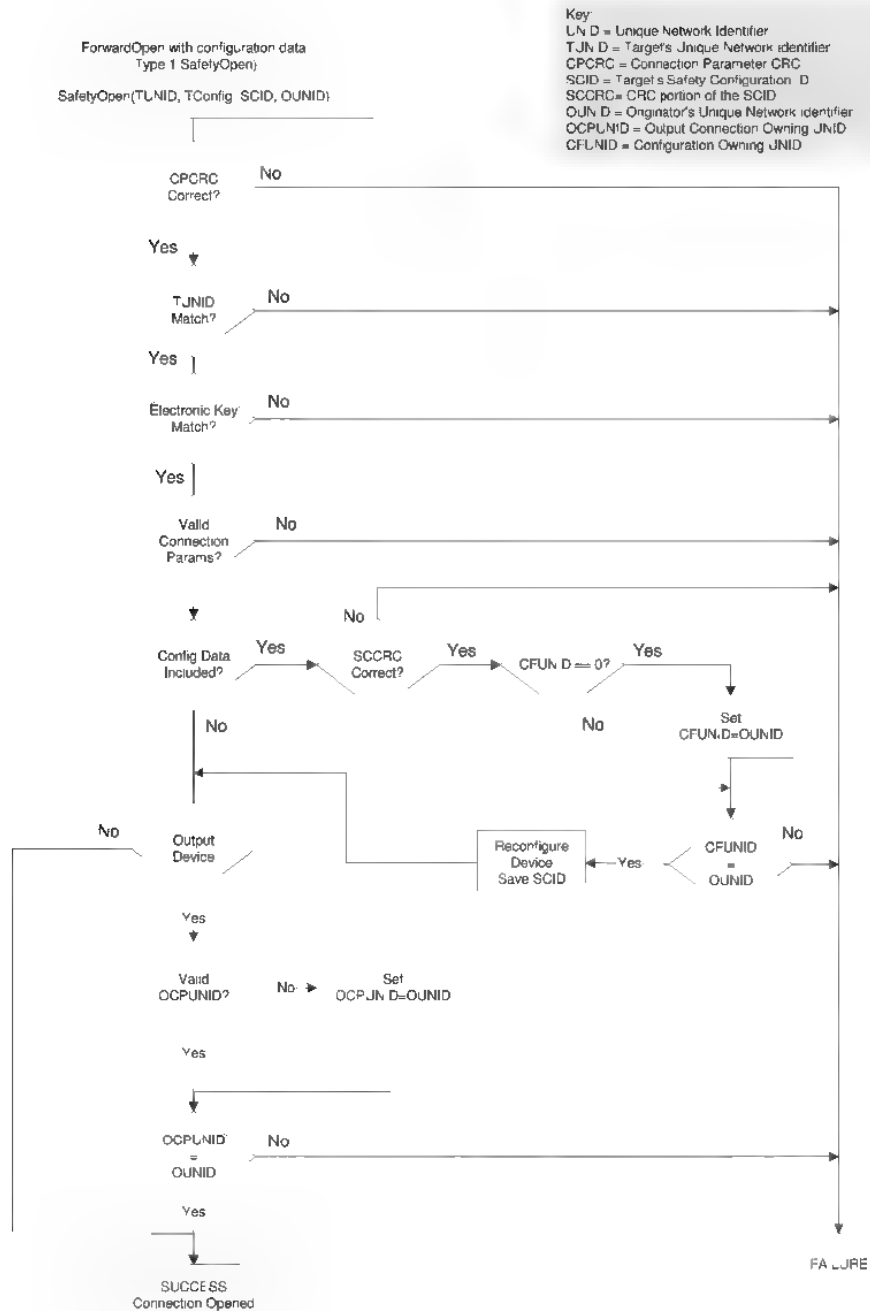




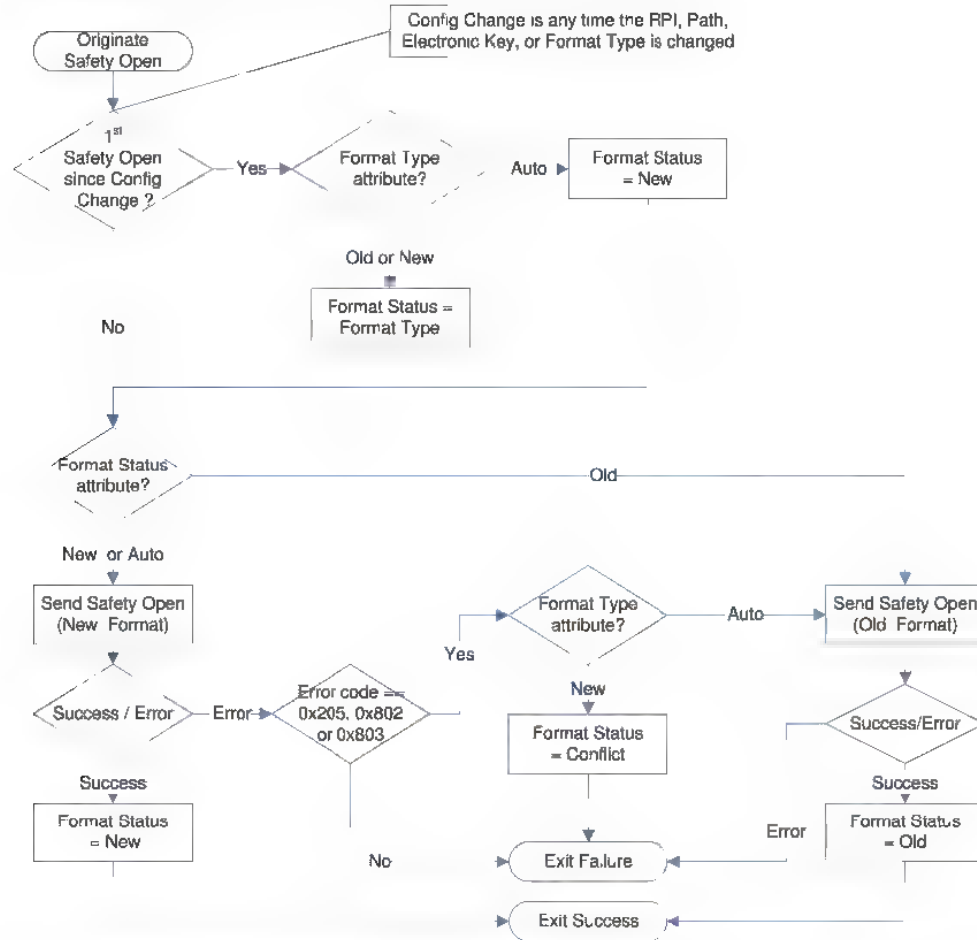
Figure 3-3.2 Target Processing for SafetyOpen with configuration data (Type 1 SafetyOpen)



Safety Originators that do not support any other means to determine which format to connect with can implement the following logic to dynamically determine the format.



Figure 3-3.3 Originator logic to determine which format to use



See Section 5-6.2.6 Connection Configuration Object, Format Type Attribute for more information.



### 3-3.5 Checks required by Multi-cast Producers with existing connections

There is a distinct possibility that the safety parameters used in a Forward\_Open or SafetyOpen from various consuming originators will not always be the same. A consistent set of rules must be used for how a Multi-cast producing target should evaluate these parameters when they conflict with other already established consuming connections.

**Table 3-3.2 Multi-Cast Producer Parameter Evaluation Rules**

Safety Parameter	Producer Evaluation Rule if connection exists	Error Response <sup>1</sup> (if appropriate)	
		General	Extended
Ping_Interval EPI Multiplier	Shall match existing P.I.E.M.	0x01 0x01	0x0801 0x0012
Time Coord Msg Min Multiplier	Accept any (all can be different)	N/A	
Network Time Expectation Multiplier	Accept any (all can be different)	N/A	
Timeout_Multiplier	Accept any (all can be different)	N/A	
Max_Consumer_Number	Shall match existing Max_Consumer_Number	0x01 0x01	0x0801 0x002A
T→O RPI, O→T RPI, and Time Correction RPI	Shall match existing connection RPIs	0x01	0x0111
Network Segment Type	Shall match existing connection format <sup>2</sup>	0x01	0x803
Max_Fault_Number	Accept any (all can be different)	N/A	

<sup>1</sup> At the option of the device developer, another connection could also be opened rather than return an error response

<sup>2</sup> Subsequent connection requests on an existing multi-cast connection must all be the same format as the existing connection.

### 3-3.6 Electronic Key Usage for Safety

Safety originators and targets shall meet the following requirements for use of the Electronic Key segment (See Table C-1.7 of The CIP Networks Library, Volume 1, Common Industrial Protocol (CIP)) within a Safety Open.

FRS341 The Electronic Key segment shall be present in all SafetyOpen or Safety Forward\_Open service requests.

FRS342 The Electronic Key shall always be processed by Target devices with integrity (cannot be confirmed externally)

FRS344 The compatibility bit and its behavior in the Electronic Key shall be supported as defined by safety originators and targets, wildcards (fields set to 0) for individual fields within the Electronic Key shall not be supported

### 3-3.7 RPI vs. API in Safety Connections

Safety devices do not have the ability to negotiate the RPI value. Safety Targets either accept the requested RPI because it can be supported, or it rejects it completely. The tolerance around an acceptable RPI request should be +/- 128uS.

FRS345 The T-to-O API, O-to-T API, and Time Correction API in a success response shall always be the same value received in the SafetyOpen



### 3-3.8 Application Path Construction for Safety

CIP Specification Volume 1 (section 3-5.5.1.9) defines a method to construct an application path in a SafetyOpen for 3 references: one for configuration, one for consumed data and one for produced data.

The possible path types are identified as follows:

- A. - [Class] [Assembly] [Instance] [Config-instance]
- B. - [Class] [Assembly] [Instance] [Producer-instance]
- C. - [Class] [Assembly] [Instance] [Consumer-instance]
- D. - [Class] [Assembly] [Instance] [Device-defined NULL instance)]
- E. - [CP] [Consumer CP]
- F. - [CP] [Producer CP]
- G. - [CP] [Device-defined NULL CP]

When all 3 paths referenced the same class, they are often formatted in a compressed format: (CP stands for Connection Point).

[Class] [Class #] [Instance] [Instance #] [CP] [Consumer CP] [CP] [Producer CP]

Since all three paths must reference the same class, a NULL instance value must be reserved in that class (by the device developers) to use the compressed form for safety. The NULL value will be used for the portion of the path that references the Time Coordination connection and configuration connection (in non-configuring originators). If a common class reference is not practical, the long form should be used.

The following rules apply to SafetyOpen path construction:

- All devices shall support the long form at a minimum
- A Null path or Null connection point shall be used to represent the Time Coordination connection.
- A Null path or Null connection point shall be used for the configuration path when no configuration data can ever be sent on the connection.
- On any connections, which can accept a Type 1 SafetyOpen, Originators shall always send a valid configuration path even if no configuration data is present (i.e. Type 2a).

The “Null” path or connection point used shall be one of the module’s choosing. The method of making this selection in an EDS file will be via the Data Path field in the Connection Manager section. Devices configured by configuration applet also must provide this path.

Safety device designers can choose the path format to specify in their EDS files from the following options. The possible three-path combinations for Type 1, 2a SafetyOpen on connections, which will accept configuration are:

- [A, D, B] for the input connection,
- [A, G, F] compressed format for input connections
- [A, C, D] for the output connection.
- [A, E, G] compressed format for output connections



The possible three-path combinations for Type 2a, 2b SafetyOpen on connections, which will not accept configuration data are:

- [D, D, B] for the input connection, and
- [D, G, F] compressed format for input connections
- [D, C, D] for the output connection.
- [D, E, G] compressed format for output connections

### **3-3.9 Safety Validator Connection Types**

The Safety Validator function, depending on the safety connection type provided in the SafetyOpen, will always require between 2-3 connection resources. The SafetyOpen contains the connection information to allow a target to set up these connections appropriately. This section will show how the connection types are determined by how the SafetyOpen must be filled out.

One basic assumption is that connections for Time Coordination and Time Correction messages do not require a path entry since they have an implied path to the Safety Validator object coordinating the safety connection. The various connection scenarios are described in the Validator object in Chapter 5.

The Safety Validator function can determine which connection resources will be required by looking at the parameter in the SafetyOpen. The various SafetyOpen parameters as well as the key parameters that the safety function will need to determine the types of connections is outlined in Table 3-3.3. Connection object instances may not be available due to target identifier capacity or limitations; there may be no unallocated connection identifiers (particularly on DeviceNet targets). Or the device may not have the processing throughput to handle the additional connection; etc. Safety Connection objects requires resources similar to standard I/O connections.



Table 3-3.3 Forward Open Setting Options for Safety Connections

O to T Connection Type	T to O Connection Type	Transport Type/ Trigger	App Path 1	App Path 2 <sup>1</sup>	App Path 3 <sup>1</sup>	Max Consumer Number	Safety Validator Connection Type	Key Decision Factors	Targets Role
Point to Point	Point to Point	Application, Class 0, Client	Vendor defined Null Path <sup>2</sup>	Vendor defined Null Path	Safety Input App. Object	1	Single Cast Producer (Input)	2 PtP + Client	<b>Producing Connection:</b> Safety Data <b>Consumer Connection:</b> Coordination
Point to Point	Point to Point	Application, Class 0, Server	Vendor defined Null Path	Safety Output App. Object	Vendor defined Null Path	1	Single-Cast Consumer (Output)	2-PtP + Server	<b>Consuming Connection:</b> Safety Data <b>Producing Connection:</b> Coordination
Point to Point	Multicast	Application, Class 0, Client	Vendor defined Null Path	Vendor defined Null Path	Safety Input App. Object	2-15	Multi-Cast Producer <b>DeviceNet</b>	2 – Multicast + 1 PTP + Client	<b>Producing Connection:</b> Safety Data <b>Producing Connection:</b> Correction <b>Consuming Connection:</b> Coordination
Point to Point	Multicast	Application, Class 0, Client	Vendor defined Null Path	Vendor defined Null Path	Safety Input App. Object	2-15	Multi Cast Producer <b>Non-DeviceNet</b>	1 Multicast + 1 PTP + Client	<b>Producing Connection:</b> Safety Data + Correction <b>Consuming Connection:</b> Coordination
Multicast	Point to Point	Application, Class 0, Server	Vendor defined Null Path	Safety Output App. Object	Vendor defined Null Path	2-15	Multi-Cast Consumer <b>DeviceNet</b>	2 – Multicast + 1 PTP + Server	<b>Consuming Connection:</b> Safety Data <b>Consuming Connection:</b> Correction <b>Producing Connection:</b> Coordination
Multicast	Point to Point	Application, Class 0, Server	Vendor defined Null Path	Safety Output App. Object	Vendor defined Null Path	2-15	Multi cast Consumer <b>Non-DeviceNet</b>	1 Multicast + 1 PTP + Server	<b>Consuming Connection:</b> Safety Data + Correction <b>Producing Connection:</b> Coordination

<sup>1</sup> Input & Output with respect to the network

<sup>2</sup> Devices configurable via EDS must define an application Null Path in the EDS file.



Table 3-3.4 shows what network connection parameters to use for all three connections associated with each safety connection type.

**Table 3-3.4 Network Connection Parameters for Safety Connections**

Target Type	Network Connection Parameter	Proper Parameter Value	Description
Single-cast Producer	O-to-T	0x4406	Single-cast High priority, fixed
	T-to-O	0x4408	Single-cast High priority, fixed
	Time Correction	0x0000	Not used
Single-cast Consumer	O-to-T	0x4408	Single-cast High priority, fixed
	T-to-O	0x4406	Single-cast High priority, fixed
	Time Correction	0x0000	Not used
Multi-cast Producer	O-to-T	0x4406	Single-cast High priority, fixed
	T-to-O	0x240E	Multi-cast High priority, fixed
	Time Correction	0x2406	Multi-cast High priority, fixed

FRS174 Successful SafetyOpen responses shall return the connection identifiers used, as well as the Consumer\_Number assigned to the originator (always 0xFFFF for single-cast connections).

### 3-3.10 Application Reply Data in a Successful SafetyOpen Response

The format for a SafetyOpen response for Success is the same as the Forward\_Open\_Reply as defined in CIP Specification Volume 1, Chapter 3.

The target of a SafetyOpen shall return the following data in the Application Reply field of the SafetyOpen response. The consumer number and PID reply data is only of significance when the originator is the consumer.

**Table 3-3.5 CIP Safety Target Application Reply (Size: 10 Bytes)**

Parameter Name <sup>1</sup>	Data Type	Description
Consumer Number	UINT	
PID/CID	STRUCT of.	Target Identifier used to generate a PID or CID seed value.
Target Vendor Id	UINT	Target's Vendor Id
Target Device Serial Number	UDINT	Target's Device Serial Number
Target Connection Serial Number	UINT	Connection Serial Number Target assigned to the connection. The Safety Validator instance Id should be used.

FRS363 When the Target of a connection using the extended format on EtherNet/IP; it shall use the 14-byte extended format SafetyOpen Target Application reply.



**Table 3-3.6 – CIP Safety Extended Format Target Application Reply**

Parameter Name <sup>1</sup>	Data Type	Description
Consumer Number	UINT	
PID/CID	Struct of:	Target Identifier used to generate a PID or CID seed value.
Target Vendor Id	UINT	Target's Vendor Id
Target Device Serial Number	UDINT	Target's Device Serial Number
Target Connection Serial Number	UINT	Connection Serial Number Target assigned to the connection. The Safety Validator instance Id should be used.
Initial Time Stamp <sup>2</sup>	UINT	Used by Consumer to initialize Last Time Stamp for Rollover
Initial Rollover Value <sup>2</sup>	UINT	Used by Consumer to initialize TS_Rollover_Count

<sup>2</sup> Consuming targets should echo back the values received in the SafetyOpen

**Table 3-3.7 SafetyOpen Target Application Reply (Size: 18 bytes)**

Parameter Name <sup>1</sup>	Data Type	Description
Consumer_Number	UINT	
PID/CID	STRUCT of:	Target Identifier used to generate a PID or CID seed value.
Target Vendor Id	UINT	Target's Vendor Id
Target Device Serial Number	UDINT	Target's Device Serial Number
Target Connection Serial Number	UINT	Connection Serial Number Target assigned to the connection. The Safety Validator instance Id should be used.
Time_Correction_Connection_ID	UDINT	Connection ID selected by the Target for the Time Correction connection. If chosen by the originator, then the value is echoed back.
Time_Correction_API	UDINT	Returns same value received in the Request packet.

<sup>1</sup> These fields are always present in the Application Reply field. When a DeviceNet target receives a SafetyOpen that is not multicast (thus no Time Correction connection is established) the Time\_Correction\_Connection\_ID parameter shall be set to 0xFFFF. If the target is not a multicast producer, the consumer number is set to 0xFFFF

FRS362 When the Target of a connection using the extended format on DeviceNet, it shall use the 22-byte extended format SafetyOpen Target Application reply.



**Table 3-3.8 SafetyOpen Reply – for Extended Format DeviceNet Producers**

Parameter Name <sup>1</sup>	Data Type	Description
Consumer Number	UINT	
PID/CID	Struct of:	Target Identifier used to generate a PID or CID seed value.
Target Vendor Id	UINT	Target's Vendor Id
Target Device Serial Number	UDINT	Target's Device Serial Number
Target Connection Serial Number	UINT	Connection Serial Number Target assigned to the connection. The Safety Validator instance Id should be used.
Time_Correction_Connection_ID	UDINT	Connection ID selected by the Target for the Time Correction connection. If chosen by the originator, then the value is echoed back.
Time_Correction_API	UDINT	Returns same value received in the Request packet.
Initial Time Stamp <sup>2</sup>	UINT	Used by Consumer to initialize Last_Time_Stamp_for_Rollover
Initial Rollover Value <sup>2</sup>	UINT	Used by Consumer to initialize TS_Rollover_Count

<sup>1</sup> These fields are always present in the Application Reply field. When a DeviceNet target receives a SafetyOpen that is not multicast (thus no Time Correction connection is established) the Time\_Correction\_Connection\_ID parameter shall be set to 0xFFFF. If the target is not a multicast producer, the consumer number is set to 0xFFFF

<sup>2</sup> Consuming targets should echo back the values received in the SafetyOpen

### 3-3.11 Unsuccessful SafetyOpen Response for Safety

#### 3-3.11.1 Safety-Specific Error Responses

The SafetyOpen has some special error cases that don't map to the existing set of error codes, and extends the definition of existing error codes

Table 3-3.9 defines new extended error codes for safety and shows the extended definition of existing error codes.

**Table 3-3.9 New and extended error codes for safety**

General Status	Extended Status	Explanation
0x01	0x0105	Ownership Conflict or OUNID Mismatch. The configuration is already owned by another originator.
0x01	0x0106	Ownership Conflict or OUNID Mismatch. The output connection was already owned by another originator.
0x01	0x0205	Parameter Error in Unconnected Send Service or Parameter Error in SafetyOpen or SafetyClose
0x01	0x801	Incompatible Multi-cast RPI An existing connection has been established at a different RPI.
0x01	0x802	Invalid Safety Connection Size
0x01	0x803	Invalid Safety Connection Format
0x01	0x804	Invalid Time Correction Connection Parameters
0x01	0x805	Invalid Ping Interval EPI Multiplier
0x01	0x806	Time Coordination Msg Min Multiplier
0x01	0x807	Network Time Expectation Multiplier



General Status	Extended Status	Explanation
0x01	0x808	Timeout Multiplier
0x01	0x809	Invalid Max Consumer Number
0x01	0x80A	Invalid CPCRC
0x01	0x80B	Time Correction Connection Id Invalid
0x01	0x80C	SCID Mismatch. The SCID was non-zero and did not match the value in the target
0x01	0x80D	TUNID not set. Device is out-of-box and TUNID has not been set, so connections are not allowed.
0x01	0x80E	TUNID Mismatch. The TUNID provided does not match. The message was likely routed to this node in error.
0x01	0x80F	Configuration operation not allowed

Event/Error Matrix Table 3-3.10 provides guidance for which error codes to use for possible errors detected in processing a SafetyOpen. This is not intended to be an exhaustive list.

**Table 3-3.10 SafetyOpen Error Event Guidance Table**

Event	Error Type	General Status	Extended Status
The target device received a SafetyOpen request in states of Safety Supervisor object other than Idle and Executing.	Configuration operation not allowed	0x01	0x080F
	Device not configured	0x01	0x110
Safety Connection Type (name) Invalid	Safety Parameter Error – Invalid Connection Type	0x01	0x315
Safety Connection Size Invalid	Safety Parameter Error – Invalid Safety Connection Size	0x01	0x0802
Safety Connection Format Invalid	Safety Parameter Error – Invalid Safety Connection Format	0x01	0x0803
Time Correction Connection Parameters Invalid	Safety Parameter Error – Invalid Time Correction Connection Parameters	0x01	0x0804
Ping Interval EPI Multiplier was out of the range	Safety Parameter Error – Invalid Ping Interval EPI Multiplier	0x01	0x0805
Time Coordination Message Min Multiplier was out of the range	Safety Parameter Error – Time Coordination Msg Min Multiplier	0x01	0x0806
Time Expectation Multiplier was out of the range	Safety Parameter Error – Time Expectation Multiplier	0x01	0x0807
Timeout Multiplier was out of the range	Safety Parameter Error – Timeout Multiplier	0x01	0x0808
Max Consumer Number not within the limits in Multi-cast Producer	Safety Parameter Error – Invalid Max Consumer Number	0x01	0x0809
CPCRC value in SafetyOpen and calculated CPCRC value.	Safety Parameter Error – Invalid CPCRC	0x01	0x080A



Event	Error Type	General Status	Extended Status
Time Correction Connection Id Invalid	Safety Parameter Error – Time Correction Connection Id Invalid	0x01	0x080B
RPI requested is incompatible with previously established multi-cast connections	Incompatible Multi-cast RPI	0x01	0x801
O→T RPI, T→O RPI, or Time Correction RPI is not supported	RPI not supported	0x01	0x0111
All Safety Validator Instances are being used.	Connection Manager or connection object cannot support any more connections.	0x01	0x0113

### **3-3.12 Forward\_Close for Safety**

There are no additional behaviors defined for the Forward\_Close service when used to close a Safety connection. In the following Safety Requirements, the term “Safety Close” refers to both the Forward\_Close of the Connection Manager object and the SafetyClose of the Connection Object

FRS334 The SafetyClose Service shall be used to close connections with a Safety Targets (and in the case of a Forward\_Close all other nodes in the connection path).

The SafetyClose request shall remove a connection from all the nodes participating in the original connection.

FRS335 The SafetyClose request shall cause all resources in all nodes participating in the connection to be deallocated, including connection IDs, bandwidth, and internal memory buffers.

FRS337 Targets receiving the SafetyClose service request shall close the connection who’s Originator Vendor ID, Connection Serial Number, and Originator Serial Number parameters match. Additional information provided by this service (ie, Connection Path) shall be ignored by the target.

A SafetyClose Success reply shall be returned after the connection has been deleted at the target.

FRS339 When a SafetyClose success reply is received, the originator and each intermediate node along the path, closes the connection and releases resources associated with that connection. But the originator shall also close the connection if no response is received within a vendor selected wait period



This page is intentionally left blank



## **Volume 5: CIP Safety**

# **Chapter 4: Object Model**

---



## **Contents**

4-1	Object Model .....	3
-----	--------------------	---



## **4-1 Object Model**

This chapter of the CIP Safety Networks specification contains additions to the Object Model that are safety specific. At this time, no such additions exist.

All Safety objects created shall have the word “Safety” as the first word of the object name.



This page is intentionally left blank



## **Volume 5: CIP Safety**

# **Chapter 5: Object Library**

---



## Contents

5-1	Introduction.....	7
5-2	Identity Object .....	8
5-2.1	Changes to Common Services .....	8
5-3	DeviceNet Object.....	9
5-3.1	DeviceNet Object Changes .....	9
5-3.2	Quick Connect Restriction for Safety .....	9
5-4	Safety Supervisor Object .....	10
5-4.1	Safety Supervisor Class Attributes.....	11
5-4.1.1	Subclasses.....	11
5-4.2	Safety Supervisor Instance Attributes .....	11
5-4.2.1	Semantics.....	14
5-4.2.1.1	Manufacturer Name .....	14
5-4.2.1.2	Software Revision Level.....	15
5-4.2.1.3	Hardware Revision Level .....	15
5-4.2.1.4	Manufacturer's Serial Number .....	15
5-4.2.1.5	Device Status .....	15
5-4.2.1.6	Exception Status .....	15
5-4.2.1.7	Exception Detail Alarm and Exception Detail Warning .....	16
5-4.2.1.7.1	Common Exception Detail.....	17
5-4.2.1.7.2	Common Exception Detail Attribute Values. ....	17
5-4.2.1.7.3	Device Exception Detail .....	17
5-4.2.1.7.4	Manufacturer Exception Detail .....	18
5-4.2.1.8	Alarm Enable and Warning Enable .....	18
5-4.2.1.9	Time .....	18
5-4.2.1.10	Scheduled Maintenance Expiration Timer.....	19
5-4.2.1.11	Scheduled Maintenance Expiration Warning Enable.....	19
5-4.2.1.12	Configuration Lock.....	19
5-4.2.1.13	Configuration UNID (CFUNID).....	19
5-4.2.1.14	Safety Configuration Identifier (SCID) .....	20
5-4.2.1.15	Target UNID (TUNID).....	21
5-4.2.1.16	Proposed TUNID (OUNID).....	21
5-4.2.1.17	Output Connection Owner (OCPUNID).....	21
5-4.2.2	Subclasses.....	22
5-4.3	Safety Supervisor Common Services .....	22
5-4.4	Safety Supervisor Object Specific Services .....	22
5-4.4.1	Recover Service.....	23
5-4.4.2	Perform_Diagnostics Service .....	24
5-4.4.3	Configure_Request Service .....	24
5-4.4.4	Validate_Configuration Service .....	25
5-4.4.5	Set Password Service.....	27
5-4.4.6	Reset_Password Service .....	28
5-4.4.7	Configuration_Lock/Unlock Service.....	28
5-4.4.8	Mode Change Service.....	29
5-4.4.9	Safety_Reset Service .....	29
5-4.4.10	Propose_TUNID Service .....	31
5-4.4.11	Apply_TUNID Service.....	32
5-4.5	Safety Supervisor Behavior.....	34
5-4.5.1	Safety Supervisor Object States.....	34
5-4.5.2	Safety Supervisor State Event Matrix .....	36
5-4.5.3	Effect of Locking on Device Behavior .....	38
5-4.5.4	State Mapping of Safety Supervisor Object to Identity Object .....	39
5-4.5.5	Safety Supervisor Event to Identity Object Event Mapping.....	40
5-4.5.6	Identity Object Event to Safety Supervisor Event Mapping.....	41



5-5	Safety Validator Object .....	42
5-5.1	Revision History .....	42
5-5.2	Class Attributes .....	42
5-5.2.1	Safety Connection Fault Count.....	42
5-5.3	Instance Attributes .....	43
5-5.3.1	Safety Validator State.....	46
5-5.3.2	Safety Validator Type.....	46
5-5.3.3	Ping Interval EPI Multiplier .....	47
5-5.3.4	Time Coord Msg Min Multiplier .....	48
5-5.3.5	Network Time Expectation Multiplier.....	48
5-5.3.6	Timeout Multiplier .....	48
5-5.3.7	Max Consumer Number .....	48
5-5.3.8	Data Connection Instance... ..	48
5-5.3.9	Coordination Connection Instance .....	49
5-5.3.10	Correction Connection Instance .....	49
5-5.3.11	CCO Binding.....	49
5-5.3.12	Max Data Age.. ..	49
5-5.3.13	Producer/Consumer Fault Counter .....	49
5-5.4	Class Services .....	50
5-5.5	Instance Services.....	50
5-5.5.1	Get_Attributes_All Response .....	50
5-5.6	Object Behavior .....	50
5-5.6.1	State Transition Diagram.....	51
5-5.6.1.1	IDLE .....	51
5-5.6.1.2	Initializing .....	52
5-5.6.1.3	Established.....	52
5-5.6.1.4	Connection_Failed.....	52
5-5.6.2	State Event Matrix.....	52
5-6	Connection Configuration Object .....	54
5-6.1	Class Attribute Extensions .....	54
5-6.2	Instance Attributes Additions and Extensions.....	54
5-6.2.1	Instance Attribute Semantics Extensions or Restrictions for Safety.....	57
5-6.2.1.1	Connection Flags – (Attribute 2) .....	57
5-6.2.1.2	CS Data Index Number - (Attribute 4).....	58
5-6.2.1.3	Connection Timeout Multiplier – (part of Attribute 5).....	58
5-6.2.1.4	Transport Class and Trigger – (part of Attribute 5) .....	58
5-6.2.1.5	O=>T RPI – (part of Attribute 5).....	59
5-6.2.1.6	O=>T Connection Parameters – (part of Attribute 5).....	59
5-6.2.1.7	T=>O RPI - (part of Attribute 5).....	60
5-6.2.1.8	T=>O Connection Parameters – (part of Attribute 5) .....	60
5-6.2.1.9	Connection Path – (Attribute 6).....	61
5-6.2.1.10	Proxy Config Data Size – (part of Attribute 7).....	61
5-6.2.1.11	Proxy Config Data – (part of Attribute 7).....	61
5-6.2.1.12	Target Config Data Size – (part of Attribute 10) .....	61
5-6.2.1.13	Target Config Data – (part of Attribute 10).....	61
5-6.2.1.14	Data Map – (Attribute 9).....	62
5-6.2.1.14.1	Format 0 Usage for Safety Scanners .....	62
5-6.2.1.14.2	Format 1 Usage for Safety Scanners .....	63
5-6.2.1.15	Proxy Device ID .....	63
5-6.2.1.16	Connection Disable (Attribute 12).. ..	63
5-6.2.2	Special Safety Related Parameters – (Attribute 13) ..	63
5-6.2.2.1	Ping Interval EPI Multiplier .....	63
5-6.2.2.2	Time Coord Msg Min Multiplier .....	64
5-6.2.2.3	Network Time Expectation Multiplier.....	64
5-6.2.2.4	Timeout Multiplier.....	64
5-6.2.2.5	Max Consumer Number.....	64



5-6.2.2.6	Target Connection UNID.....	65
5-6.2.2.7	Safety Configuration CRC (SCCRC) .....	65
5-6.2.2.8	Configuration Signature (Time Stamp).....	66
5-6.2.2.9	Time Correction EPI.....	66
5-6.2.2.10	Time Correction Connection Parameters .....	66
5-6.2.3	Connection Parameter CRC (CPCRC) – (Attribute 14) .....	66
5-6.2.4	Configuration Instance – (Attribute 15).....	67
5-6.2.5	Id Allocation.....	67
5-6.2.6	Format Type .....	67
5-6.2.7	Format Status.....	69
5-6.3	Object-Specific Services .....	70
5-6.4	Common Service Extensions for Safety.....	70
5-6.4.1	Get Attribute All (Service Code 0x01) .....	70
5-6.4.2	Set_Attribute_All (Service Code 0x02).....	72
5-6.4.3	Restore (Service Code 0x15).....	72
5-6.5	Object Behavior .....	72
5-7	Safety Discrete Input Point (SDIP).....	74
5-7.1	Revision History .....	74
5-7.2	Class Attributes.....	74
5-7.3	Instance Attributes .....	74
5-7.4	Common Services .....	76
5-7.5	Object-specific Services.....	76
5-7.6	Behavior.....	77
5-7.6.1	Type of Safety Inputs .....	77
5-7.6.2	Safety Discrete Input Evaluation Behavior .....	78
5-8	Safety Discrete Input Group (SDIG) .....	79
5-8.1	Revision History .....	79
5-8.2	Class Attributes .....	79
5-8.3	Instance Attributes (Common) .....	80
5-8.4	Common Services .....	80
5-8.5	Object-specific Services.....	80
5-8.6	Behavior.....	80
5-9	Safety Discrete Output Point (SDOP).....	81
5-9.1	Revision History .....	81
5-9.2	Class Attributes .....	81
5-9.3	Instance Attributes (Common).....	81
5-9.4	Common Services .....	82
5-9.5	Object-specific Services.....	82
5-9.6	Behavior.....	82
5-10	Safety Discrete Output Group (SDOG) .....	84
5-10.1	Revision History.....	84
5-10.2	Class Attributes .....	85
5-10.3	Instance Attributes (Common) .....	85
5-10.4	Common Services .....	85
5-10.5	Object-specific Services .....	86
5-10.6	Behavior .....	86
5-11	Safety Dual Channel Output Object (SDCO) .....	87
5-11.1	Revision History.....	87
5-11.2	Class Attributes .....	87
5-11.3	Instance Attributes (Common) .....	87
5-11.4	Common Services .....	88
5-11.5	Object-specific Services.....	88
5-11.6	Dual Channel Evaluation Behavior.....	88
5-12	Discrete Output Point Object.....	90
5-12.1	Test Output Mode for Safety Attribute .....	90
5-12.2	Test Output Mode Semantic.....	90



5-13	TCP/IP Object.....	91
5-13.1	TCP/IP Object Instance Attribute Changes.....	91
5-14	Safety Analog Input Point Object.....	92
5-14.1	Revision History.....	92
5-14.2	Class Attributes.....	92
5-14.3	Instance Attributes.....	92
5-14.4	Semantics.....	96
5-14.4.1	Data Type, Attribute 1.....	96
5-14.4.2	Safety Analog Input Value, Attribute 2.....	96
5-14.4.3	Input Range, Attribute 3.....	97
5-14.4.4	Input Channel Mode, Attribute 4.....	97
5-14.4.5	Input Point Status, Attribute 5.....	98
5-14.4.6	Point Fault Reason, Attribute 6.....	98
5-14.4.7	5-11.5.7 Real Time Sample Period, Attribute 7.....	98
5-14.4.8	5-11.5.8 Input Error Latch Time, Attribute 8.....	99
5-14.4.9	Full Scale, Attribute 9.....	99
5-14.4.10	Safe State Behavior, Attribute 10.....	100
5-14.4.11	Safe State Value, Attribute 11.....	100
5-14.4.12	High and Low Signal, High and Low Engineering Values, Attributes 12-15.....	100
5-14.4.13	Alarm/Warning Status, Attributes 16.....	100
5-14.4.14	Alarm/Warning Enable, Attributes 17, 22.....	101
5-14.4.15	Alarm/Warning Trip Points, Hysteresis and Settling Time, Attributes 18-21, 23-26.....	101
5-14.4.16	Over Range Value, Attribute 27.....	102
5-14.4.17	Under Range Value, Attribute 28.....	102
5-14.4.18	Offset-A and Gain Data Type, Attributes 29, 31.....	103
5-14.4.19	Offset-A and B, Gain and Unity Gain Reference, Attributes 30, 32 - 34.....	103
5-14.4.20	Produce Trigger Delta, Attribute 35.....	103
5-14.4.21	Subclass, Attribute 99.....	103
5-14.5	Common Services.....	103
5-14.5.1	Get_Attributes_All Response.....	104
5-14.5.2	Set_Attributes_All Request.....	106
5-14.6	Object-specific Services.....	107
5-14.6.1	Zero Adjust and Gain Adjust Services.....	107
5-14.7	Behavior.....	108
5-14.7.1	Type of Safety Inputs.....	108
5-14.7.2	Safety Analog Input Evaluation.....	109
5-15	Safety Analog Input Group Object.....	110
5-15.1	Revision History.....	110
5-15.2	Class Attributes.....	110
5-15.3	Instance Attributes.....	111
5-15.4	Semantics.....	112
5-15.4.1	Number of Bound Instances, Attribute 1.....	112
5-15.4.2	Bound Instances, Attribute 2.....	112
5-15.4.3	Data Type, Attribute 3.....	113
5-15.4.4	Input Range, Attribute 4.....	113
5-15.4.5	Input Status, Attribute 5.....	113
5-15.4.6	Input Error Latch Time, Attribute 6.....	113
5-15.4.7	Full Scale, Attribute 7.....	113
5-15.4.8	Safe State Behavior, Attribute 8.....	113
5-15.4.9	Safe State Value, Attribute 9.....	113
5-15.4.10	Alarm and Warning Status, Attribute 10.....	113
5-15.4.11	Alarm/Warning Enable, Attributes 11, 16.....	114
5-15.4.12	Alarm/Warning Trip Points, Hysteresis and Settling Time, Attributes 12-15, 17-20.....	114
5-15.5	Common Services.....	114
5-15.5.1	Get_Attributes_All Response.....	115
5-15.5.2	Set_Attributes_All Request.....	116



5-15.6	Object specific Services .....	117
5-15.7	Behavior .....	117
5-15.7.1	Attribute Access Rules .....	118
5-15.7.2	Input Status Attribute.....	118
5-15.7.3	Alarm and Warning Status Attribute ... ..	118
5-16	Safety Dual Channel Analog Input Object.....	119
5-16.1	Revision History.....	119
5-16.2	Class Attributes .....	119
5-16.3	Instance Attributes.....	119
5-16.4	Semantics .....	120
5-16.4.1	Dual Channel Mode, Attribute 1 .....	120
5-16.4.2	Dual Channel Evaluation Status, Attribute 2.....	121
5-16.4.3	Discrepancy Timer, Attribute 3 ....	121
5-16.4.4	Member One, Attribute 4.....	121
5-16.4.5	Member Two, Attribute 5.....	121
5-16.4.6	Discrepancy Deadband, Attribute 6.....	121
5-16.5	Common Services .....	122
5-16.5.1	Get_Attributes_All Response .....	122
5-16.5.2	Set_Attributes_All Request .....	123
5-16.6	Object-specific Services .....	123
5-16.7	Behavior .....	124
5-16.7.1	Safety Dual Channel Input Evaluation .....	124
5-16.7.2	Discrepancy Evaluation.....	125
5-16.7.3	Dual Channel Evaluation.....	126



## **5-1 Introduction**

This section identifies and defines new safety objects that are specified for use by CIP safety devices. This section also defines the extensions of existing CIP objects.



## 5-2 Identity Object

### Class Code: 01<sub>hex</sub>

This section describes changes to the Identity object to support CIP Safety.

SRS142 All CIP safety devices shall replace the Identity object state behavior with the behavior defined in the Safety Supervisor object.

### 5-2.1 Changes to Common Services

Table 5-2.1 Identity Object Common Service Changes

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
05 <sub>hex</sub>	Optional	Conditional*	Reset	Invokes the Reset Service for the Device

\*Conditional – SRS61 The Identity object Reset service shall not be supported in safety devices (reply to any requests with “Service Not Supported” error). The service shall be replaced by the Reset service defined by the Safety Supervisor.



## 5-3 DeviceNet Object

**Class Code: 03<sub>hex</sub>**

### 5-3.1 DeviceNet Object Changes

This section defines the changes and additions required of the DeviceNet object to support CIP safety. An attribute shall be added to allow the Safety Network Number (SNN) to be read. The definition for this attribute is as follows:

**Table 5-3.1 New CIP Safety-specific DeviceNet Object Instance Attribute**

Attrib ID	Need in Implem	Access Rule	NV	Name	DeviceNet Data Type	Description of Attribute	Semantics of Value
11	Required	Get	NV	Safety Network Number	6 octets	System-wide unique number for the subnet this device resides on.	Attribute shall contain the SNN portion of the TUNID in the Safety Supervisor. See Chapter 2-3.1.1 for a complete definition of the SNN.

SRS102 Devices implementing DeviceNet Safety shall support attributes 11 of the DeviceNet object.

### 5-3.2 Quick Connect Restriction for Safety

All CIP Safety Device shall not support the Quick Connect feature. The Quick connect attribute shall not be supported and shall respond to any Get/Set attribute service with the error code "Attribute not supported".



## **5-4 Safety Supervisor Object**

### **Class Code: 39 hex**

The Safety Supervisor is the core object of CIP Safety Devices. There are two fundamental Safety Supervisor implementations defined:

1. Baseline Supervisor – defined as a supervisor implementation that supports all the “required” attributes and services defined by this object definition. The baseline Supervisor includes all state machine behavior excluding state event processing for services not required (refer to Section 5-4.5.1).
2. SNCT Supervisor – defined as a supervisor implementation that supports all the Baseline functionality plus the attributes and services that are marked “conditional” on SNCT support.

SRS65 All safety devices on a CIP safety network shall support the baseline Safety Supervisor definition.

Because many of the parameters in the safety supervisor object are deemed safety critical, the attributes of this object shall have a higher integrity than standard object data.

SRS66 The state behavior of the Safety Supervisor object shall control the state of all CIP objects in a device. All CIP objects in a safety device are required to be subservient to the states and commands of the Safety Supervisor to manage its functions and behaviors.

The Safety Supervisor object centralizes application object state definitions and related status information, exception status indications (alarms and warnings), and defines a behavior model which is assumed by objects identified as belonging to safety devices. That is, if a reset is requested of the Safety Supervisor object instance, it is performed by this object instance as well as all of its associated application objects.

The Identity object shall get its state information from the Safety Supervisor object. A reset request to the Identity object is not supported by CIP Safety devices (refer to Section 5-2 and shall be replaced by a reset request to the Safety Supervisor object which has additional qualifiers to execute the request. Further relationships are specified in the behavior section below.

Additionally, some instance attributes are reserved to allow for future extension to implement profiles defined for the semiconductor industry.



## 5-4.1 Safety Supervisor Class Attributes

**Table 5-4.1 Safety Supervisor Class Attributes**

Attribute ID	Need in implementation	Access Rule	Name	Date Type	Description of Attribute
1 thru 96	These class attributes are described in Chapter 4 of CIP Common (Volume 1)				
97-98	Reserved by CIP for Future Use				
99	Conditional *	Get	Subclass	UINT	Identifies a subset of additional class attributes, services, and behaviors

\* If the value of Subclass is 00 which identifies "no subclass", then this attribute is **OPTIONAL** in implementation, otherwise, this attribute is **REQUIRED**.

### 5-4.1.1 Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID 63<sub>hex</sub> and numbers downward for services. The subclass for a given class is identified by the value of its subclass class attribute. There are presently no subclasses defined for the class instance.

## 5-4.2 Safety Supervisor Instance Attributes

**Table 5-4.2 Safety Supervisor Instance Attributes**

Attr ID	Need in implementation	Access Rule	NV	Name	Date Type	Description of Attribute
1	Optional	Get	NV	Number of Attributes	USINT	Number of Attributes supported by the object instance
2	Optional	Get	NV	Attribute List	Array of USINT	List of attributes supported by the object instance
3	Reserved					
4	Reserved					
5	Optional	Get	NV	Manufacturer Name	Short String	ASCII Text, Max. 20 characters, see Semantics Section
6	Optional	Get	NV	Manufacturer Model Number	Short String	ASCII Text, Max. 20 characters, Manufacturer specified, see Semantics Section
7	Optional	Get	NV	Software Revision Level	Short String	ASCII Text, Max. 6 characters, see Semantics Section
8	Optional	Get	NV	Hardware Revision Level	Short String	ASCII Text, Max. 6 characters, see Semantics Section
9	Optional	Get	NV	Manufacturer Serial Number	Short String	ASCII Text, Max. 30 characters, Manufacturer specified, see Semantics Section
10	Optional	Get	NV	Device Configuration	Short String	ASCII Text, Max. 50 Characters, Manufacturer Specified. Optional additional information about the device configuration
11	Required	Get	V	Device Status	USINT	See "Semantics" Section



Attr ID	Need in implementation	Access Rule	NV	Name	Date Type	Description of Attribute
12	Required	Get	V	Exception Status	BYTE	See “Semantics” Section
13	Conditional based on Exception Status Bit 7	Get	V	Exception Detail Alarm	STRUCT of:	A Structure of 3 structures containing a bit-mapped representation of the alarm detail
	Common Exception Detail			STRUCT of:		
	Size			USINT	Number of Common Detail Bytes	
	Detail			ARRAY of:	See “Semantics” Section	
	Detail n			BYTE	See “Semantics” Section	
	Device Exception Detail			STRUCT of:		
	Size			USINT	Number of Device Detail Bytes	
	Detail			ARRAY of:	See Device Profile	
	Detail n			BYTE	See Device Profile	
	Manufacturer Exception Detail			STRUCT of:		
	Size			USINT	Number of Manufacturer Detail Bytes	
	Detail			ARRAY of:	Manufacturer Specified	
	Detail n			BYTE	Manufacturer Specified	
14	Conditional Based on Exception Status Bit 7	Get	V	Exception Detail Warning	STRUCT of:	A Structure of 3 structures containing a bit-mapped representation of the warning detail
	Common Exception Detail			STRUCT of:		
	Size			USINT	Number of Common Detail Bytes	
	Detail			ARRAY of:	See “Semantics” Section	
	Detail n			BYTE	See “Semantics” Section	
	Device Exception Detail			STRUCT of:		
	Size			USINT	Number of Device Detail Bytes	
	Detail			ARRAY of:	See Device Profile	
	Detail n			BYTE	See Device Profile	
	Manufacturer Exception Detail			STRUCT of:		
	Size			USINT	Number of Manufacturer Detail Bytes	
	Detail			ARRAY of:	Manufacturer Specified	
	Detail n			BYTE	Manufacturer Specified	



Attr ID	Need in implementation	Access Rule	NV	Name	Date Type	Description of Attribute
	Conditional Based on Size			Size	USINT	Number of Manufacturer Detail Bytes
				Detail	ARRAY of:	Manufacturer Specified
				Detail n	BYTE	Manufacturer Specified
15	Required	Set	NV	Alarm Enable	BOOL	See “Semantics” Section
16	Required	Set	NV	Warning Enable	BOOL	See “Semantics” Section
17	Optional	Set	<sup>1</sup>	Time	DATE AND TIME	The value of the Device’s internal real-time clock.
18	Conditional – Required if (Attribute 17) is supported	Get	NV	Clock Power Cycle Behavior	USINT	Describes the behavior of the device’s internal real-time clock (the “Time” attribute) during a power cycle  0 = [default] clock always resets during power cycle  1 = clock value is stored in nonvolatile memory at power down  2 = clock is battery-backed and runs without device power  3-255 – Reserved
19	Optional	Get	NV	Last Maintenance Date	DATE	The date on which the device was last serviced
20	Optional	Get	NV	Next Scheduled Maintenance Date	DATE	The date on which it is recommended that the device next be serviced
21	Optional	Get	NV	Scheduled Maintenance Expiration Timer	INT	See “Semantics” Section
22	Conditional – Required if Calibration Expiration is supported	Set	NV	Scheduled Maintenance Expiration Warning Enable	BOOL	See “Semantics” Section
23	Optional	Get	NV	Run Hours	UDINT	An indication of the number of hours that the device has had power applied.
24	Conditional <sup>3</sup>	Set only by “Configuration Lock/Unlock” Service  Gettable	NV	Configuration Lock	BOOL	Marks the device configuration contained within as Verified and locked, or alternatively as unverified, unlocked, and changeable.



Attr ID	Need in implementation	Access Rule	NV	Name	Date Type	Description of Attribute
25	Conditional <sup>3</sup>	Get	NV	Configuration UNID	10 octets	CFUNID - Identifies the owner of a Device Configuration. all 0xFF = Tool-only configuration. 0 = un-owned, accept any owner
26	Required	Get	NV	Safety Configuration Identifier	10 octets	The SCID is comprised of the Safety Configuration CRC + Safety Configuration Time Stamp. This is the signature for the Configuration
27	Required	Get	NV	Target UNID	10 octets	The current UNID of the device.
28	Conditional <sup>4</sup>	Get	NV	Output Connection Point Owners		
				Number of Array Entries	UINT	Number of OCPUNID struct entries
				Output Owners	ARRAY of STRUCT	
				OCPUNID	10 octets	The owner UNID for the output resource 0 = un-owned, accept any owner (default)
				ePath Size	USINT	Path size, number of bytes
				Application Resource	Packed ePath	The path to owned resource (i.e. 20 04 24 01 - Assembly class, instance 1)
29	Conditional <sup>3</sup>	Get	V	Proposed TUNID	10 octets	The UNID value that an originator is attempting to set in the device.
97 98	Reserved by CIP for Future Use					
99	Conditional <sup>2</sup>	Get	NV	Subclass	UINT	Identifies a subset of additional instance attributes, services, and behaviors

1. The nonvolatile behavior of the Time attribute is conditional on the value of the Clock Power Cycle Behavior attribute.
2. If the value of Subclass is 00 which identifies "no subclass", then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED.
3. Safety Devices which support the SNCT interface (i.e. not configured by the Safety Open or some vendor specific means) shall support this attribute.
4. Conditional SRS135 the Output Connection Owner attribute shall be supported for all safety output assemblies.

## 5-4.2.1 Semantics

### 5-4.2.1.1 Manufacturer Name

The Manufacturer Name attribute, identifies the manufacturer of the device.



The Manufacturer Name attribute is not guaranteed, by specification, to be unique. Therefore, it is not a substitute for the corresponding attribute of the Identity object and should not be used for identification purposes.

#### **5-4.2.1.2 Software Revision Level**

This is an ASCII coded text string representing the revision of the software corresponding to the specific device identified by the Identity object.

#### **5-4.2.1.3 Hardware Revision Level**

This is an ASCII coded text string representing the revision of the hardware which is identified by the Identity object. The manufacturer of the device must control this revision such that modifications to the device hardware may be tracked.

#### **5-4.2.1.4 Manufacturer's Serial Number**

This attribute is a string representation of the vendor's serial number of the device, formatted to fit most manufacturing tracking systems. It is not required that this attribute match the Identity object's serial number which is used to uniquely identify the device in the network environment.

#### **5-4.2.1.5 Device Status**

This attribute represents the current state of the device. Its value changes as the state of the device changes. The following values are defined:

**Table 5-4.3 Device Status Attribute State Values**

Attribute Value	State
0	Undefined
1	Self-Testing
2	Idle
3	Self-Test Exception
4	Executing
5	Abort
6	Critical Fault
7	Configuring
8	Waiting for TUNID
9-50	Reserved by CIP
51-99	Device Specific
100-255	Vendor Specific

#### **5-4.2.1.6 Exception Status**

A single byte attribute whose value indicates that the status of the alarms and warnings for the device. This indication may be provided in one of two methods: Basic or Expanded.



For the *Basic Method*, bit seven of the Exception Status attribute is set to zero; all exceptions are reported exclusively through communication of this Exception Status attribute. The format of bits zero through six in this mode is device specific; the format may be further specified in an appropriate device profile specification; if it is not specified, then the format of bits zero through six is equivalent to that specified for the expanded method.

For the *Expanded Method*, bit seven of Exception Status attribute is set to one; exceptions are reported through the communication of this Exception Status attribute, formatted as specified in the table below. In addition, the Exception Detail attributes are supported. The Exception Status bits are determined by a logical “OR” of the related Exception Detail bits, as indicated.

**Table 5-4.4 Exception Status Attribute Format**

Exception Status Bit Map, Bit 7 Set to 0		Exception Status Bit Map, Bit 7 set to 1
Bit	Function	Function
0	Device Specific definition (See Device Profile)	ALARM/Device-common*
1		ALARM/Device-specific
2		ALARM/Manufacturer-specific
3		Reserved set to 0
4		WARNING/device-common*
5		WARNING/device-specific
6		WARNING/manufacturer-specific
7	0 = Basic Method	1 = Expanded Method

\* The alarm or warning is not specific to the device type or device type manufacturer.

#### 5-4.2.1.7 Exception Detail Alarm and Exception Detail Warning

The formats of these two attributes are identical. Therefore, they are described together here:

Attributes that relate the detailed status of the alarms or warnings associated with the device. Each attribute is a structure containing three members; these three members respectively relate the detailed status of exceptions that are common (i.e., not device-specific), device-specific but not manufacturer-specific, and manufacturer-specific. The common detail is defined below. The device-specific detail is defined in the appropriate Device Profile. The manufacturer-specific detail is defined by the manufacturer. A SIZE value of zero indicates that no detail is defined for the associated exception detail structure.

Each of the three structure members is defined as a structure containing an ordered list (i.e., array) of bytes of length SIZE, and an unsigned integer whose value is SIZE. Each of the bytes in each array has a specific mapping. This mapping is formatted as 8 bits representing 8 independent conditions, whereas a value of 1 indicates that the condition is set (or present), and a value of 0 indicates that the condition is cleared (or not present). Note that if a device does not support an exception detail, the corresponding bit is never set. The bitmaps for alarms and warnings in the corresponding attributes are structured in parallel so that a condition may have either alarm or warning set depending on severity. If a condition inherently cannot be both alarm and warning, then the parallel bit position corresponding to the other state shall remain “0.”



The existence of an exception detail variable structure is dependent on the value of the Exception Status Attribute; the existence of an exception detail variable structure is only required if bit seven of the Exception Status attribute is set to 1 (indicating Expanded method reporting) and the bit (among bits zero through six) of the Exception Status attribute corresponding to the particular exception type is also set to 1.

#### 5-4.2.1.7.1 Common Exception Detail

This structure relates exception conditions (i.e., alarms or warnings) which are common to all devices. The Detail element of the structure is an ordered list (i.e., array) of bytes of length [SIZE] which is the value of the structure element Size. For each byte in the Detail field, all bits which are not identified are reserved for future standardization.

The first byte in this attribute is CommonExceptionDetail[0]. Additional exception details, if provided, are named CommonExceptionDetail[1], . . . CommonExceptionDetail[SIZE]. The specific exception associated with each of the bitmaps is given in the table below. The SIZE for this revision is two (2). The criteria details for each exception condition are outside the scope of this document.

#### 5-4.2.1.7.2 Common Exception Detail Attribute Values

Table 5-4.5 Common Exception Detail Attribute Values

Common Exception Detail [0] *	Common Exception Detail [1] *
Internal diagnostic exception	power supply overcurrent
Microprocessor exception	reserved power supply
EPROM exception	power supply output voltage
EEPROM exception	power supply input voltage
RAM exception	scheduled maintenance due
Reserved by CIP	notify manufacturer
Internal real-time exception	reset exception
Reserved by CIP	Reserved by CIP

\* Note that if a device does not support an exception detail, the corresponding bit is never set

#### 5-4.2.1.7.3 Device Exception Detail

This structure, similar in form to Common Exception Detail, relates exception conditions which are specific to individual devices on the network and are defined in their respective device profiles. The Detail element of the structure is an ordered list (i.e., array) of bytes of length [SIZE] which is the value of the structure element size. For a detailed description of this attribute, consult the appropriate specific device profile.



**5-4.2.1.7.4 Manufacturer Exception Detail**

This structure, similar in form to Common Exception Detail, relates exception conditions which are specific to the manufacturers of individual devices on the network and are defined by them in their product documentation. The Detail element of the structure is an ordered list (i.e., array) of bytes of length [SIZE] which is the value of the structure element Size. For a detailed description of this attribute, consult the appropriate specific device manufacturer documentation

**Table 5-4.6 Exception Detail Format Summary**

Detail Component	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Common Exception Detail Size	0	0	0	0	0	0	0	0
Common Exception Detail 0	Reserved	Real-time Fault	Reserved	Data Memory	Non-Volatile Memory	Code Memory	Micro-processor	Diagnostic
Common Exception Detail 1	Reserved	Reset Exception	Notify Vendor	Scheduled Maint. Due	PS Input Voltage	PS Output Voltage	Reserved	PS Over Current
Device Exception Detail Size	x	x	x	x	x	x	x	x
Device Exception Detail 0	--	--	--	--	--	--	--	--
Device Exception Detail n	--	--	--	--	--	--	--	--
Manufacturer Exception Detail Size	x	x	x	x	x	x	x	x
Manufacturer Exception Detail 0	--	--	--	--	--	--	--	--
Manufacturer Exception Detail n	--	--	--	--	--	--	--	--

**5-4.2.1.8 Alarm Enable and Warning Enable**

These Boolean attributes are used to enable (1) or disable (0) the Safety Supervisor object's process of setting Exception bits. When disabled, corresponding bits are never set; and, if they were set, disabling clears them. Also, alarm and warning states are not retained; when enabled, bits shall be set only if the corresponding condition is true.

The default state for these Enable attributes is enabled (1).

**5-4.2.1.9 Time**

This optional attribute represents the value of the time and date as maintained by the device's real-time clock with a resolution of one millisecond.



The default value for the Time attribute is zero (0), corresponding to 12:00AM, January 1, 1972, as specified by Appendix C of CIP Common.

#### **5-4.2.1.10 Scheduled Maintenance Expiration Timer**

This attribute, with a resolution of one hour, is used to cause a warning which indicates that a device calibration is due. An Safety Supervisor timer decrements this attribute once per hour while power is applied. When the attribute is no longer positive and the Scheduled Maintenance Expiration Warning Enable attribute is set to enabled, a Scheduled Maintenance Expiration Warning condition is generated. This causes the Scheduled Maintenance Due Warning bit to be set.

The Scheduled Maintenance Expiration Timer attribute, when implemented, shall not wrap; when the attribute reaches its most negative value, it no longer decrements. The Scheduled Maintenance Expiration Timer, when implemented, shall continue to decrement irrespective of the state of the Scheduled Maintenance Expiration Warning Enable attribute. The Scheduled Maintenance Expiration Timer, when implemented, shall be maintained in nonvolatile memory.

#### **5-4.2.1.11 Scheduled Maintenance Expiration Warning Enable**

This Boolean attribute is used to enable (1) or disable (0) the Safety Supervisor object's process of setting the Scheduled Maintenance Due Exception bit. When disabled, the corresponding bit is never set; and, if it was set, disabling clears it. When enabled, the bit shall be set only if the corresponding condition is true.

The default state for this Enable attribute is enabled (1).

#### **5-4.2.1.12 Configuration Lock**

SRS68 The Configuration Lock attribute, when implemented with the SNCT interface, shall be used to mark the device configuration as verified and to lock down the configuration; 0 = Unlocked (unverified and changeable), 1 = Locked (verified and not changeable).

The default value for the Configuration Lock attribute is 0 = unlocked. Its primary use is for safety devices that are configured by the SNCT (i.e. not configured by the safety open). After the tool has configured the device and the user has validated the configuration, the software sets this flag to lock out any further changes. It is an option that this flag be protected by password which requires user verification before the configuration can be unlocked and changed.

SRS199 While a device has the Configuration Lock attribute set, it shall reject Configure Requests, Type 1 & 2 Safety Reset, and any valid Type 1 SafetyOpen.

#### **5-4.2.1.13 Configuration UNID (CFUNID)**

The Configuration UNID attribute shall be implemented as part of the SNCT interface. The Configuration UNID identifies the owner of the configuration in this safety device. The Configuration UNID attribute can be set by a number of means.



SRS70 The Configuration UNID attribute shall have the following special meanings assigned;

- All octets set to 0x00 = No owner, accept any
- All octets set to 0xFF = Software Tool is the assigned owner

SRS134 The default for the Configuration UNID shall be all octets set to 0x00 .

SRS205 When the Configuration UNID attribute is set to “No owner, accept any”, CIP Safety Devices shall capture the first configuration source (Type 1 and/or Configure\_Request) as the owner. (refer to Table 5-4.7).

The CFUNID value is used in conjunction with the OUNID value provided in either the Configure\_Request or a Type 1 SafetyOpen. See Table 5-4.7 for how devices respond to the various services based on what the CFUNID is set to at the time the service is received. For example, if the CFUNID is set to all 0xFF, then the OUNID of the Configure\_Request must equal this value to process the command.

SRS71 SNCT shall always use all 0xFF for the OUNID.

**Table 5-4.7 Summary of Device Behavior for various CFUNID values**

Service	Value of CFUNID		
	0xFF	0x00	OUNID
Configure_Request (0xFF) *from SNCT	Accept	Accept Set CFUNID=0xFF	Fail
Configure_Request (OUNID)	Fail	Accept Set CFUNID=OUNID	Accept only if CFUNID=OUNID Fail otherwise
Type 1 SafetyOpen	Fail	Accept Set CFUNID=OUNID	Accept only if CFUNID=OUNID Fail otherwise

#### **5-4.2.1.14 Safety Configuration Identifier (SCID)**

The Safety Configuration Identifier attribute is the signature on the safety configuration. This attribute is a concatenation of the Safety Configuration CRC (SCCRC) + Safety Configuration Time Stamp (SCTS). These values are obtained from either the Safety Open or from the Safety Configuration Software during configuration validation.

SRS184 The default for SCID attribute in the Safety Supervisor shall be 0 .

SRS198 When a device enters Configuring mode, the SCID attribute shall be set to 0 and maintained at this value (through power cycles) until a successful Validate has been executed.

```
typedef struct _S_SCID
{
    DWORD      dwCRC; //calculated by device
    DWORD      dwTime; // set by software
    WORD       wDate;

} S_SCID;
```



#### **5-4.2.1.15 Target UNID (TUNID)**

The Target UNID is an NVS attribute that reflects the UNID saved with the configuration.

SRS107 The Target UNID attribute in the Safety Supervisor shall always reflect either the default or the current Target UNID saved through the UNID setting process.

SRS116 The UNID of a device shall be stored to NVS as part of the UNID setting operation.

SRS191 The UNID of a device shall be stored to the DeviceNet, TCP/IP or SERCOS III object's Safety\_Network\_Number attribute as part of the UNID setting operation.

SRS117 A device in Manufacturers default state shall have an invalid UNID value in its NVS (e.g. 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF).

SRS118 The UNID of a device shall be read from NVS during power up and loaded into an attribute of the Safety Supervisor object and Safety\_Network\_Number attribute of the DeviceNet, TCP/IP or SERCOS III object.

SRS119 If the UNID represents a valid Number (not equal to out-of-box), this number shall be compared to the MACID or IP Address. In case of a match the device continues its startup procedure. In case of a mismatch the Device transits to Abort.

SRS120 When a device faults from a Switch-NVS setting mismatch, it shall allow either a reset to out-of-box, or a Recover request issued to the Safety Supervisor to clear the Abort. In both cases, the device shall move to the "Waiting for TUNID" state.

SRS124 The reading of the UNID attribute in the Safety Supervisor shall be done with safety integrity.

SRS125 The applying of configuration data and UNID to NVS shall be done with safety integrity.

#### **5-4.2.1.16 Proposed TUNID (OUNID)**

The default value for the Proposed TUNID attribute shall be all 0xFFs.

While a device is going through the "ProposeTUNIG/ApplyTUNID" process, this attribute reflects the value being proposed. Once the apply operation is complete, this attribute should be set back to the out-of-box default of all 0xFFs.

#### **5-4.2.1.17 Output Connection Owner (OCPUNID)**



This attribute is only used for safety connection to outputs. Safety connections to outputs define an owner to prevent errant connections from hijacking an output resource in a validated safety system. This owner Id identifies which originator safety device has been given ownership rights to this safety output connection. This parameter is obtained from the Safety Segment Parameters in the Safety Open. The OCPUNID value could be the same as Attribute 25 of the Safety Supervisor (CFUNID) when the device is also configured as part of the Safety Open. Since device may have more than one output resource and may allow for separate owners of each, this attribute is arranged as a structure to allow for each resource to be separately maintained. The resource is identified by a Logical Segment address.

SRS201 The default for all OCPUNIDs shall be 0 (accept any owner) with the ePath for each pointing to the respective resource.

### 5-4.2.2 Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96hex and numbers downward for attributes, and ID 63hex and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute. There are no subclasses currently defined for the instances.

### 5-4.3 Safety Supervisor Common Services

Table 5-4.8 Safety Supervisor Common Services

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
0E <sub>hex</sub>	Conditional	Required	Get_Attributes_Single	Returns the contents of the specified attribute
010 <sub>hex</sub>	n/a	Required	Set_Attributes_Single	Modifies an attribute value
0D <sub>hex</sub>	n/a	Conditional <sup>1</sup>	Apply	Applies a pending configuration to NV memory, and moves the device to the Idle state.
15 <sub>hex</sub>	n/a	Optional	Restore	Restores last applied configuration in devices that support it.

1. Safety Devices which support the SNCT interface (i.e. not configured by the Safety Open or some vendor specific means) shall support this service.

Any Safety Supervisor instance service may be requested internally by the device as specified by the manufacturer.

### 5-4.4 Safety Supervisor Object Specific Services

Table 5-4.9 Safety Supervisor Object Specific Services

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
4B <sub>hex</sub>			Reserved	
4C <sub>hex</sub>	n/a	Optional	Recover	Moves the device out of the <b>Abort</b>



Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
4E <sub>hex</sub>	n/a	Optional	Perform_Diagnostics	Causes the device to perform a set of diagnostic routines if appropriate
4F <sub>hex</sub>	n/a	Conditional <sup>1</sup>	Configure_Request	Moves the device to the <b>Configuring</b> State if appropriate. This service requires a password be provided to be executed
50 <sub>hex</sub>	n/a	Conditional <sup>1</sup>	Validate_Configuration	Causes the device to perform a validation check of the pending configuration. This service requires the software to provide the SCID (CRC + Time Stamp)
51 <sub>hex</sub>	n/a	Conditional <sup>1</sup>	Set_Password	Used to set a password attribute in the Supervisor object
52 <sub>hex</sub>	n/a	Conditional <sup>1</sup>	Configuration_Lock/Unlock	This service acts upon the Configuration Lock attribute in the Safety Supervisor. This service requires a password be provided to be executed.
53 <sub>hex</sub>	n/a	Conditional <sup>3</sup>	Mode Change	Required for Logic devices which must support “executing” independent of I/O connections. A password is required to execute mode changes in safety devices.
54 <sub>hex</sub>	n/a	Type 0 Required, Type 1 & 2 Conditional <sup>1</sup>	Safety Reset	Resets the device in a manner similar to the Identity object, except that a password is required to execute.
55 <sub>hex</sub>	n/a	Conditional <sup>2</sup>	Reset Password	Used to reset the password in a device. Data contents of this message is vendor specific
56 <sub>hex</sub>	n/a	Conditional <sup>4</sup>	Propose_TUNID	Used to initiate the setting of the TUNID in an out-of-box device
57 <sub>hex</sub>	n/a	Conditional <sup>4</sup>	Apply_TUNID	Used to complete the TUNID setting operation

1. Required for CIP Safety devices which support the SNCT interface.

2. Conditional on supporting the password service as part of the SNCT interface

3. Required for CIP Safety devices which support the SNCT interface and are Logic devices which must support “executing” independent of I/O connections. A password is required to execute mode changes in safety devices.

4. Required for CIP Safety devices which support the SNCT interface but highly recommended for devices which are configurable by Type 1 SafetyOpen but don't support the SNCT interface

Any Safety Supervisor instance service may be requested internally by the device as specified by the manufacturer. Generally, these requests will be generated as the result of an event such as the activation of a button or external contact closure.

#### 5-4.4.1 Recover Service

Used to transition the device application objects, out of the Recoverable Fault state, to the idle state. This service request may be originated internally, from application objects.



**5-4.4.2 Perform\_Diagnostics Service**

Used to instruct the Safety Supervisor object to perform a diagnostic test. A diagnostic test is either of type *common* or *device-dependent*. *Common* diagnostic tests could include: RAM, EPROM, non-volatile memory, and communications. The structure of *common* type diagnostic tests are implementation-specific. Details of *device-dependent* diagnostics is outside the scope of this document.

**5-4.4.3 Configure Request Service**

Used to transition the device application objects from the **Abort**, **Execute** or **Idle** state to the **Configuring** state.

SRS72 When a Configure\_Request is received, if the Configuration\_Lock attribute is set, an “Object State Conflict (0x0C)” error shall be returned.

The request contains three parameters; the password parameter contained in the Safety Supervisor, the Target device UNID, and the Originator UNID. These values are matched up against the Password, TUNID, and CFUNID in the device. The OUNID is required to allow the device to determine if the source is a software tool or an ACR capable originator.

SRS73 When the Configure Request command is accepted, the device shall only accept configuration messages to safety relevant objects over “this connection only”. Any attempts to write to safety-relevant objects over other connections shall be rejected.

SRS74 If a configuration connection fails for any reason, another Configure\_Request shall be received (establishing a new connection source) before configuration changes can be made (even after a power-cycle) .

The structure of the **Configure\_Request** is shown in the table below.

**Table 5-4.10 Configure\_Request Message Structure**

Name	Data Type	Description
Service Code	USINT	4F <sub>hex</sub>
Password	16 octets	
TUNID	10 octets	Target Device UNID. Used to detect misrouted requests.
OUNID	10 octets	Originator Device UNID. If the originator is a software tool, this parameter should be set to all 0xFF. If the originator is a device with ACR, this parameter should be the device's OUNID

SRS143 If a TUNID, OUNID mismatch error occurs when processing a Safety Supervisor Configure Request, an “Invalid Parameter (0x20)” error code shall be returned.

SRS147 If a password mismatch occurs when processing a Safety Supervisor Configure Request, a “Privilege Violation (0x0F)” error code shall be returned.



**5-4.4.4 Validate\_Configuration Service**

Causes the device to validate a pending configuration by executing a CRC calculation over this data and comparing it to the CRC provided in the message. If the CRC does not yield the same result, an error response as defined below is returned. Otherwise, a “success” response is returned. Note: This command can only be received over the connection that put the device into the configuration mode. The device NV saves the final SCID in Attribute 26 of the Safety Supervisor.

The structure of the **Validate\_Configuration** is shown in the table below.

**Table 5-4.11 Validate\_Configuration Message Structure**

Name	Data Type	Description
Service Code	USINT	50 <sub>hex</sub>
SCID	STRUCT of:	Safety Configuration ID
SCCRC	4 octets	Proposed Safety Configuration CRC <b>as calculated by the software</b> . This is validated by the Device
SCTS	6 octets	Safety Configuration Time Stamp. This value marks the date and time of the configuration, which is used as a change detection mechanism.

The structure of the Success **Validate Reply** is shown in the table below.

**Table 5-4.12 Validate\_Configuration Success Message Structure**

Name	Data Type	Description
Service Code	USINT	50 <sub>hex</sub>
SCID	STRUCT of:	Safety Configuration ID
SCCRC	4 octets	Safety Configuration CRC <b>as calculated by the Device</b> .
SCTS	6 octets	Safety Configuration Time Stamp. This value is an echo of what was sent in the Validate request.

SRS151 If the Safety Supervisor Validate\_Configuration command does not succeed, the error response shall send a class defined “Validation Error” (0xD0) . The following code is defined as a Validation Error in the Safety Supervisor.

**Table 5-4.13 Validate\_Configuration Error Code**

General Error Code	Additional Error Code	Error Name	Description
0xD0	See Table 5-4.14	Validation Error	This error is a class specific error for the Safety Supervisor and is returned in response to a Validate_Configuration error

**Table 5-4.14 Validate\_Configuration Extended Codes**

Additional Error Codes	Reason Code	Description
1	CRC mismatch	This is the default non-specific additional code, when no other additional information is available and the configuration CRC does not match the sent value



2	Invalid Configuration Parameter	This code can be used to indicate a parameter error if a device does specific validation of configuration data. This could happen for example even if the CRC's matched.
3	TUNID Not Set	This code can be used to indicate that the device is in the Waiting for TUNID state and needs to have its TUNID set.

**Figure 5-4.1**Applying Device Configuration

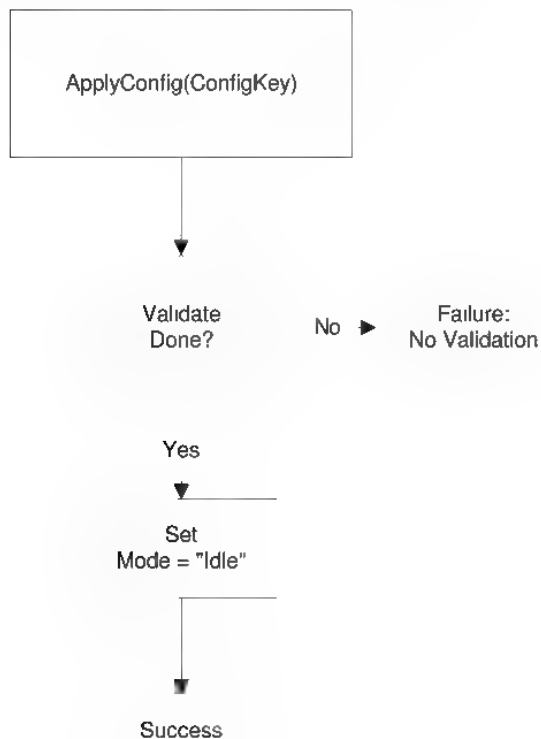
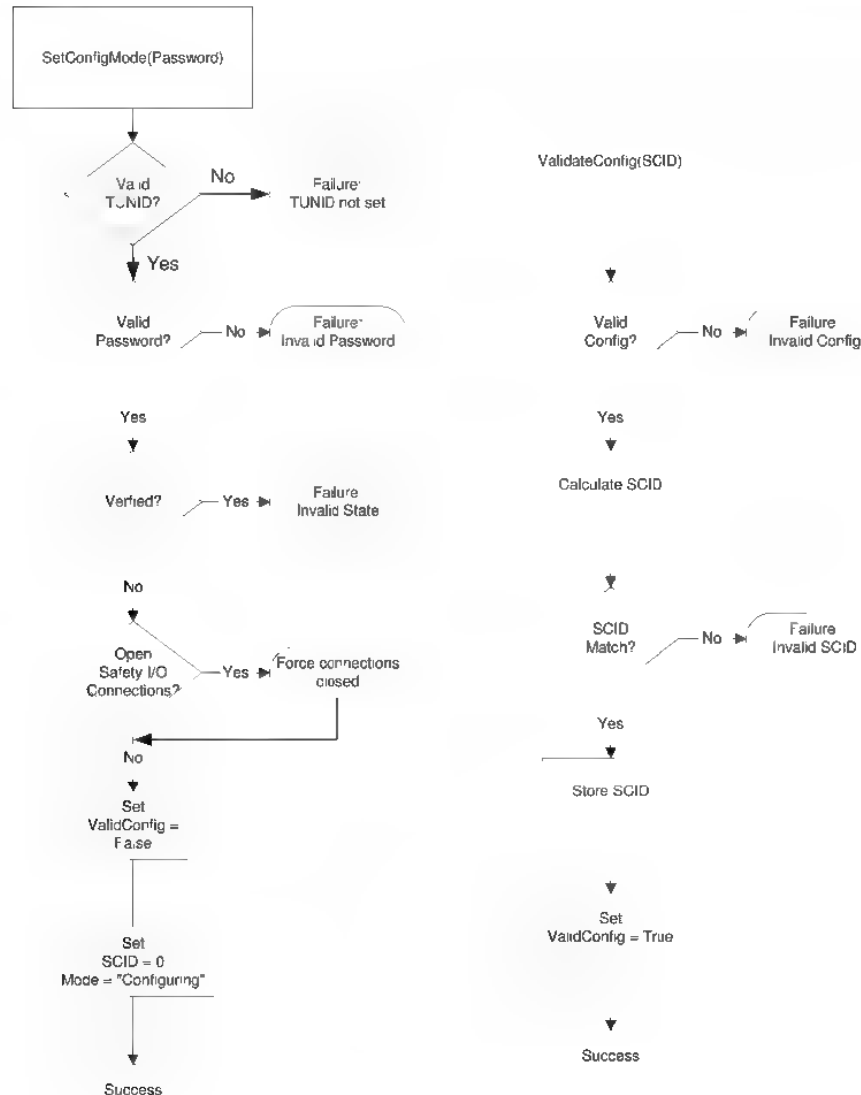




Figure 5-4.2 Configure and Validate Processing Flowcharts



#### 5-4.4.5 Set\_Password Service

Causes the device to set a Password in the Supervisor. This service takes 2 parameters. The first one is a password parameter representing the existing password, and the second is a parameter representing the new parameter. When setting a password for the first time, the Current Password parameter is set to the Out-of-box default (zero).

The structure of the **Set\_Password** is shown in the table below.



Table 5-4.15 Set\_Password Message Structure

Name	Data Type	Description
Service Code	USINT	51 <sub>hex</sub>
Current Password	16 octets	
New Password	16 octets	

SRS148 If a password mismatch occurs on the current password when processing a Safety Supervisor Set Password command, a Privilege Violation error code (0x0F) shall be returned

A password which is smaller than the password parameter should be right-aligned in little endian order to the least significant byte of the 16-octet password parameter. The unused upper bytes of the password parameter should be zero-filled.

#### 5-4.4.6 Reset\_Password Service

Provided to allow passwords in a device to be reset. The exact contents and usage has been made vendor specific to protect password integrity. This service requires 1 parameter plus a vendor specific data field. The required parameter is a data size field to identify the size of the vendor specific data field. The content of vendor data is assumed to be unique to the device and the means by which this service is used must be provided by each vendor.

The structure of the **Reset\_Password** is shown in the table below.

Table 5-4.16 Reset\_Password Message Structure

Name	Data Type	Description
Service Code	USINT	55 <sub>hex</sub>
Data Size	USINT	Number of Vendor specific data bytes
Vendor data	Data Size	Vendor data which contains data needed to cause the password to be reset

FRS346 If a Data Size error occurs in the Safety Supervisor Reset\_Password command, either error code 0x15 (too much data) or 0x13 (not enough data) shall be returned.

FRS347 If a Vendor data mismatch occurs when processing a Safety Supervisor Reset\_Password command, a Privilege Violation error code (0x0F) shall be returned.

#### 5-4.4.7 Configuration\_Lock/Unlock Service

Causes the device to modify the Configuration\_Lock Attribute in the Safety Supervisor. This service requires a value parameter (0 = unlocked, 1 = locked), and the password parameter. The password parameter is compared to the stored password and will only be executed if the password matches.

The structure of the **Configuration\_Lock/Unlock** is shown in the table below.



**Table 5-4.17 Configuration\_Lock/Unlock Message Structure**

Name	Data Type	Description
Service Code	USINT	52 <sub>hex</sub>
Value	BOOL	0 = unlocked, 1 = locked
Password	16 octets	
TUNID	10 octets	Target Device UNID. Used to detect misrouted requests.

SRS145 If a TUNID mismatch occurs in the Safety Supervisor Configuration\_Lock/Unlock command, the Invalid Parameter error code (0x020) shall be returned.

SRS149 If a password mismatch occurs when processing a Safety Supervisor Configuration\_Lock/Unlock command, a Privilege Violation error code (0x0F) shall be returned

#### 5-4.4.8 Mode Change Service

Used by devices which need to move between Idle and Executing states regardless of its connection status. This command requires proper authorization to execute, so the password is required to execute.

The structure of the **Mode Change** is shown in the table below.

**Table 5-4.18 Mode\_Change Message Structure**

Name	Data Type	Description
Service Code	USINT	53 <sub>hex</sub>
Value	BOOL	0 = Idle, 1 = Executing
Password	16 octets	

SRS144 If a password mismatch occurs in the Safety Supervisor Mode Change command, an Privilege Violation error code (0x0F) shall be returned .

#### 5-4.4.9 Safety\_Reset Service

Used to reset the device. The Safety\_Reset service replaces the standard reset service defined so that it can be qualified by the password.

SRS200 When a Type 1 or Type 2 Safety Reset is received, if the Configuration\_Lock attribute is set, the “Object State Conflict error code (0x0C)” shall be returned

The structure of the **Safety\_Reset** is shown in the table below.



**Table 5-4.19 Safety\_Reset Message Structure**

Name	Data Type	Description
Service Code	USINT	54 <sub>hex</sub>
Reset type	USINT	See Table 5-4.20
Password	16 octets	
TUNID	10 octets	Target Device UNID. Used to detect misrouted requests.
Attribute Bit Map	USINT	Bit mapped parameter which indicates which attributes should be preserved through the reset. This parameter is only included when the Reset Type is 2. See Table 5-4.21

SRS146 If a TUNID mismatch occurs when sending the Safety Supervisor Safety Reset command, the Invalid Parameter error code (0x20) shall be returned.

SRS150 If a password mismatch occurs when processing a Safety Supervisor Safety Reset command, a Privilege Violation error code (0x0F) shall be returned

**Table 5-4.20 Safety Supervisor Safety Reset Types**

Value:	Type of Reset
0	Emulate as closely as possible cycling power on the device.
1	Return as closely as possible to the default configuration, and then emulate cycling power as closely as possible.
2	Return as closely as possible to the out-of-box configuration <b>except</b> to preserve the parameters indicated by the Attribute Bit Map, and then emulate cycling power as closely as possible.

**Table 5-4.21 Attribute Bit Map Parameter**

Bit	Attribute Bit Map Assignments
0	When set, preserve Soft-set MacId
1	When set, preserve Soft-set Baud Rate
2	When set, preserve the TUNID
3	When set, preserve the Password
4	When set, preserve the CFUNID
5	When set, preserve the OCPUNID
6	Reserved, always 0
7	Use Extended Map (To be defined later)

Table 5-4.22 shows how a safety device should process the various reset types based on the Lock/Unlock condition and safety connection status.



Table 5-4.22 Reset Processing Rules for Rest Types

Reset Type	Possible Conditions when Reset is received			
	Safety Connection open / Locked	Safety Connection open / UnLocked	No Safety Connections / Locked	No Safety Connections / UnLocked
Type 0	Mode/State Error Response (SRS15)	Mode/State Error Response (SRS15)	Process & Respond	Process & Respond
Type 1	Mode/State Error Response (SRS15)	Mode/State Error Response (SRS15)	Mode/State Error Response	Process & Respond
Type 2	Mode/State Error Response (SRS15)	Mode/State Error Response (SRS15)	Mode/State Error Response	Process & Respond

Possible responses: Mode/State Error Response, Process & Respond

#### 5-4.4.10 Propose\_TUNID Service

The Propose\_TUNID service establishes a proposed TUNID setting within the device. The value isn't stored in NV memory or in the TUNID attribute until the apply service is received.

SRS194 The Propose\_TUNID service shall only be accepted when the target device is in the Waiting\_for TUNID state.

Once the TUNID is successfully set, it can transition to Configuring.

SRS195 The MacId or IP portion of the TUNID shall be matched up against the MacId of the DeviceNet or IP attribute of the TCP/IP object.

The value in the MacId attribute of the DeviceNet object or IP Attribute of the TCP/IP object will be either what was read on the switches or what was set by a tool. In either case, the MacId/IP portion of the Proposed\_TUNID must match this

The structure of the Propose\_TUNID service is shown in the table below.

Table 5-4.23 Propose TUNID Service

Name	Data Type	Description
Service Code	USINT	56 <sub>hex</sub>
TUNID	10 octets	Proposed Target Device UNID value.



SRS196 If an originator wants to cancel a propose/apply operation, sending a propose service with a TUNID of all 0xFFs shall cancel the operation.

```

TUNIDProposedHandler()
{
  IF (Device_State=Waiting_for_TUNID)
    THEN IF (Propose TUNID[TUNID] != all 0xFF)
      THEN IF (Propose_TUNID[TUNID_MacId] == MacId attribute/IP Attribute)
        THEN
          TUNIDProposedAttr = Propose_TUNID[TUNID]
          NET LED Flash sequence is started
          Return Success Reply
        ELSE
          Error Code "Invalid Parameter" is returned
      ELSE // cancel propose_apply operation, returns attribute to all 0xFF
        Stop NET LED Flash Sequence
        TUNIDProposedAttr = Propose_TUNID[TUNID]
        Return Success Reply
    ELSE
      Error Code "Object State Conflict" is returned
}

```

#### 5-4.4.11 Apply\_TUNID Service

The structure of the **Apply\_TUNID** service is shown in the table below.

**Table 5-4.24 Propose\_TUNID Service**

Name	Data Type	Description
Service Code	USINT	57 <sub>hex</sub>
TUNID	10 octets	Proposed Target Device UNID value.

SRS197 The **Apply\_TUNID** service shall validate the TUNID against the proposed value, stops the LED flashing, saves the TUNID to nonvolatile storage, updates the TUNID attribute in the Supervisor, and sets the proposed\_TUNID attribute to all 0xFFs. The Supervisor shall then transition to Configuring.

```

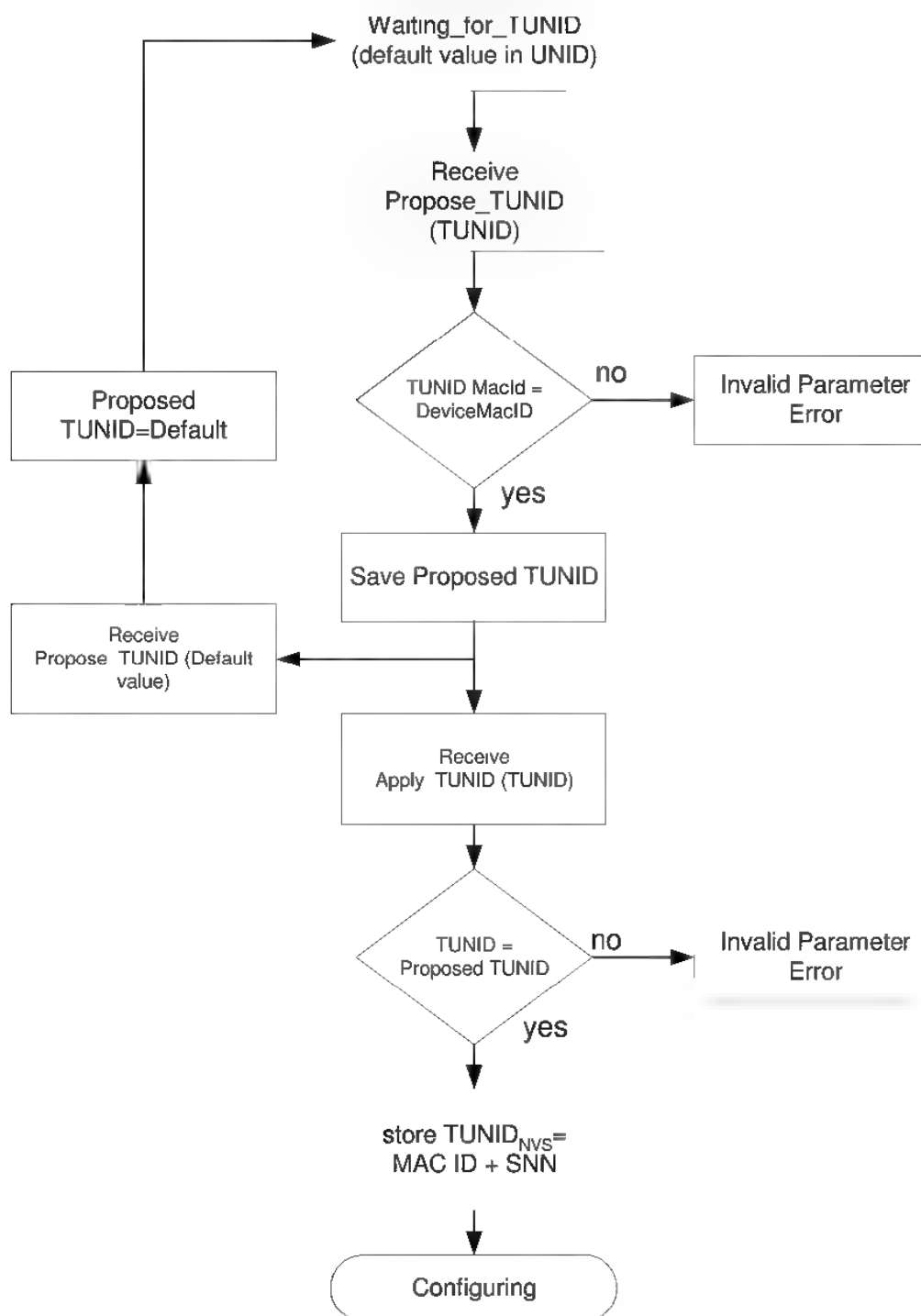
{ Apply_TUNIDHandler()
  IF (Device State=Waiting for TUNID)
    THEN IF ((Apply TUNID[TUNID] != all 0xFF) AND
      (TUNIDProposedAttr == Apply_TUNID[TUNID]))
      THEN
        Stop Flash Sequence
        Target_UNID = TUNIDProposedVar[TUNID]

//NV Memory operation
// Set SNN attribute in DeviceNet or TCP/IP object, NV Memory operation
  Safety_Network_Number = TUNIDProposedVar[SNN]
  TUNIDProposedVar[TUNID] = TUNID_OUT_OF_BOX_DEFAULT
  Return Success Reply
  Device_State = CONFIGURING
  ELSE
    Error Code "Invalid Parameter" is returned
  ELSE
    Error Code "Object State Conflict" is returned
}

```



**Figure 5-4.3 UNID Handling during “Waiting for TUNID”**

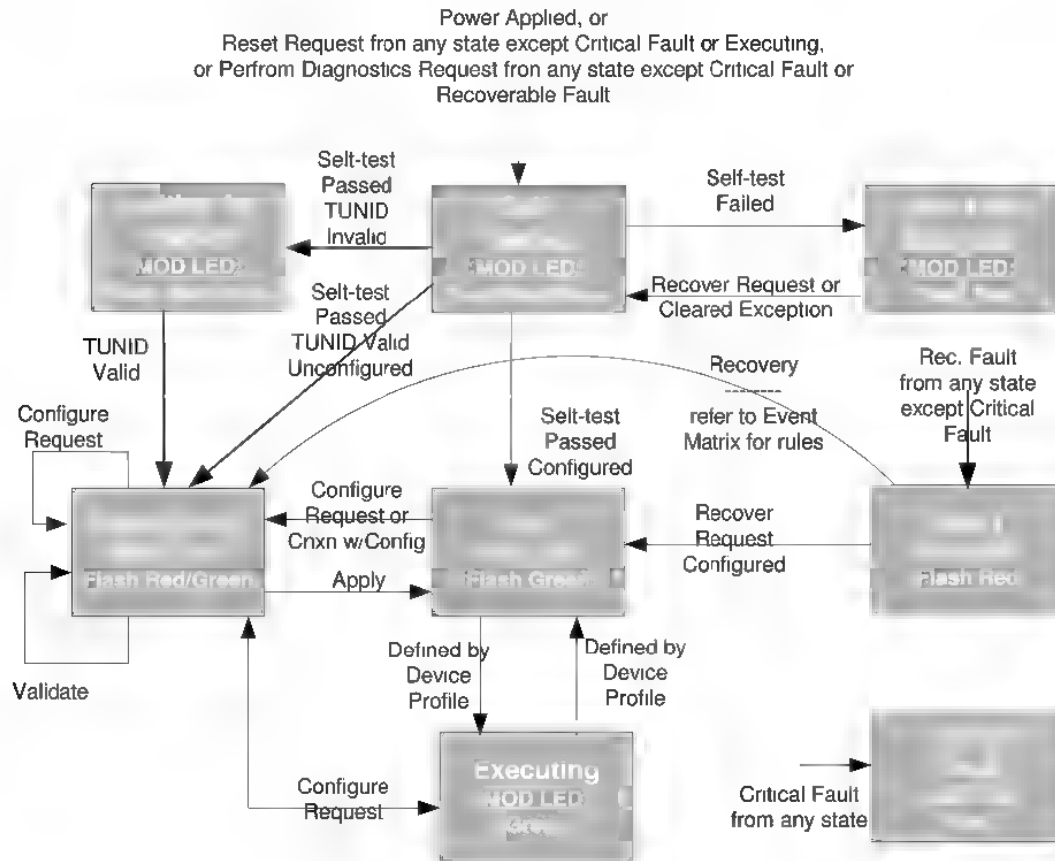




## 5-4.5 Safety Supervisor Behavior

### 5-4.5.1 Safety Supervisor Object States

Figure 5-4.4 Safety Supervisor State Diagram





**Table 5-4.25 Safety Supervisor Events**

State	Description
Executing	Device is executing, (e.g., is fully configured, has no errors, and has met vendor defined criteria). This state, and the transitions into or out of it, shall be further specified in an appropriate device profile specification.
Self-Testing	Object instance exists and has been initialized; all attributes have appropriate initial values (as indicated herein and in applicable device profile specification) Exception Status bits have been reset. Device is performing device-specific and device type specific test to determine if it is qualified to execute its application process
Self-Test Exception	Object has detected an exception condition during self-testing. The details of the exception are stored in the appropriate attribute values of the Safety Supervisor object.
Idle	Object and dev.ce have been initialized, successfully completed self-testing, and has a valid configuration. Further, the device is not executing the operational components of its device specific functions. Configuring and Idle are sticky states that are preserved through power cycles.
Abort	Object instance is in a Abort state; this state can only be entered due to internally generated events. The conditions required for exit from a recoverable fault is defined in the Event Matrix
Waiting for TUNID	Device exits Self-testing and finds the NV saved TUNID is at the out-of-box default value. It remains in this state until the TUNID setting sequence (i.e. propose/apply in SNCT interface) is successfully completed. Only then can the device be configured
Configuring	Object and dev.ce have been initialized, successfully completed self-testing, but does NOT have a valid configuration. Configuring and Idle are sticky states that are preserved through power cycles
Critical Fault	The object (and device) are in a fault state from which there is no recovery. Object services cannot be processed. The conditions required for exit from a critical fault is outside the scope of this specification.

The Safety Supervisor Status attribute value indicates the overall state of the device. It is updated on appropriate state transitions within the Safety Supervisor object. Attribute values 1 through 8 represent valid states. A value of zero indicates that the Safety Supervisor state is unknown; conditions under which a zero value may occur are outside the scope of this document.



### 5-4.5.2 Safety Supervisor State Event Matrix

**Table 5-4.26 State Event Matrix for Safety Supervisor**

Event	State							
	Waiting for TUNID	Configuring (Safety State)	Idle (Safety State)	Self Test (Safety State)	Self Test Exception (Safety State)	Executing	Abort (Safety State)	Critical Fault (Safety State)
Power Applied			--	Default Entry Point performs its Self test application process	--	--	--	Transition to SELF TEST
Self-Test Passed			Not Applicable	Transition to Last Saved State if not out-of-box Else, Wait for TUNID	Not Applicable	Not Applicable	Not Applicable	Not Applicable
Self-Test Failed			Not Applicable	Set appropriate Exception Status Bits and Transition to SELF TEST EXCEPTION	Not Applicable	Not Applicable	Not Applicable	Not Applicable
Propose TUNID	Validate Settings and Flash Net LED pattern	Error OSC**	Error OSC**	Error OSC**	Error OSC**	Error OSC**	Error OSC**	Error OSC**
Apply TUNID	Save TUNID to NV memory, Transition to Configuring	Error OSC**	Error OSC**	Error OSC**	Error OSC**	Error OSC**	Error OSC**	Error OSC**
Exception Condition Cleared			Not Applicable	Not Applicable	Set appropriate Exception Status Bits and Transition to SELF TESTING	Not Applicable	Not Applicable	Not Applicable
Critical Fault	Transition to CRITICAL FAULT	Transition to CRITICAL FAULT	Transition to CRITICAL FAULT	Transition to CRITICAL FAULT	Transition to CRITICAL FAULT	Transition to CRITICAL FAULT	Transition to CRITICAL FAULT	Ignore Event



Event	State							
Configure Request	Error TNS***	Stay in state Validate and accept	If not Locked, Transition to Configuring	Error OSC**	Error OSC**	If not Locked, Drop Connections Apply Safety State Go to Configuring	If Configured: Transition to IDLE -- Unconfigured: Transition to Configuring --- If Switch/TUNID mismatch Error OSC**	Error OSC**
Validate Configuration	Error OSC**	Stay in state Store Result	Error OSC**	Error OSC**	Error OSC**	Error OSC**	Error OSC**	Error OSC**
Apply Request	Error OSC**	Transition to Idle if Valid	Error OSC**	Error OSC**	Error OSC**	Error OSC**	Error OSC**	Error OSC**
Type 1 SafetyOpen	Error TNS***	Apply configuration Transition to Executing if Config. Valid	Transition to Configuring, Proceed to IDLE/EXECUTING dependent on Profile	Error OSC**	Error OSC**	Drop Connections, Transition to Configuring, Proceed to IDLE EXECUTING dependent on Profile	Error OSC***	Error OSC**
Reset Request	Process type Transition to SELF TEST	Process type Transition to SELF TEST	If not Locked, Process type Transition to SELF TEST	Process type Transition to SELF TEST	Process type Transition to SELF TEST	If not Locked, Process type Transition to SELF TEST	If not Locked, Process type Transition to SELF TEST	Process type Transition to SELF TEST
Internal Abort Request	Transition to Abort	Transition to Abort	Transition to Abort	Error OSC**	Error OSC**	Transition to Abort	Error AIRS*	Error OSC**
Recover Request	Error OSC**	Error OSC**	Error OSC**	Restart SELF TESTING	Transition to SELF TESTING	Error OSC**	If Configured Transition to IDLE. Unconfigured: Transition to Configuring If Switch/TUNID mismatch Error OSC**	Error OSC**
Perform Diagnostics Request	Error OSC**	Error OSC**	Transition to SELF TEST	Restart SELF TEST	Transition to SELF TEST	Error OSC**	Perform all device diagnostic tests	Error OSC**
Safety Connection Failed	Ignore Event	Ignore Event	Ignore Event	Ignore Event	Ignore Event	Device Profile Defined	Ignore Event	Ignore Event
Safety I/O Connection established	Not Possible	Not Possible	Device Profile Defined	Ignore Event	Not Possible	Device Profile Defined	Not Possible	Not Possible



Event	State							
Safety I/O Connection Deleted	Ignore Event	Ignore Event	Ignore Event	Ignore Event	Not Possible	Device Profile Defined	Ignore Event	Error OSC**
Mode Change	Error OSC**	Error OSC**	Device Profile Defined	Error OSC**	Error OSC**	Device Profile Defined	Error OSC**	Error OSC**
Restore	Error OSC**	Restore Previous Configuration	Error OSC**	Error OSC**	Error OSC**	Ignore Event	Error OSC**	Error OSC**
Lock/Unlock	Error OSC**	Error OSC**	if valid, Execute and reply	Error OSC**	Error OSC**	if valid, Execute and reply	if valid, Execute and reply	Error OSC**

**LED Definitions for States**

Module Status LED	Flashing Red/Green	Flashing Red/Green	Flashing Green	Flashing Red/Green	Flashing Red	Solid Green	Flashing Red	Solid Red
-------------------	--------------------	--------------------	----------------	--------------------	--------------	-------------	--------------	-----------

\* Error AIRS = Error Response “Already in Requested Mode/State” (Code 0x0B)

\*\* Error OSC = Error Response “Object State Conflict” (Code 0x0C) when possible

\*\*\* Error TNS= Error Response “TUNID not Set”, 0x01 extended code or 0x80D object specific extended code

Ignore Event:: defined as “event has no effect on the state”, no response necessary

Any Safety Supervisor instance service may be requested internally by the device as specified by the manufacturer. Generally, these requests will be generated as the result of an event such as a diagnostic failure.

### 5-4.5.3 Effect of Locking on Device Behavior

Figure 5-4.5 shows the effect that the “locked” condition has on the ability of a device to be configured; devices which are locked will reject all attempts to change any configuration data that affects the SCID.



Figure 5-4.5 Configuration, Testing and Locked Relationships.

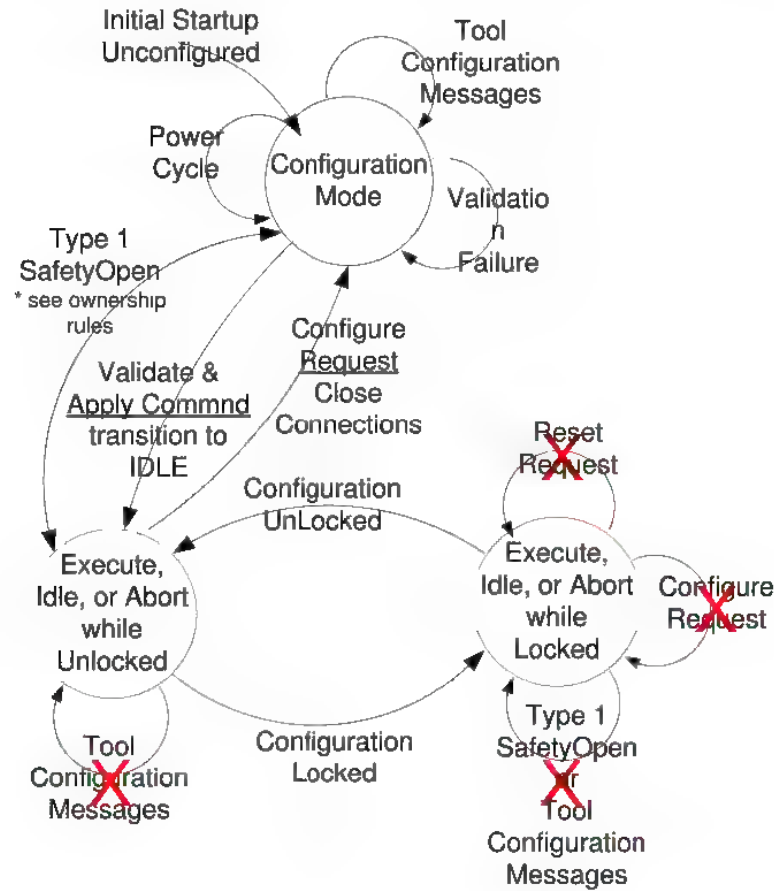


Table 5-4.27 shows the effect that ownership has on a device's ability to be configured. It shows that owned devices can only be Type 1 configured by the originator that has the owner UNID. A locked device can never be configured regardless.

Table 5-4.27 Configuration Owner Control vs. Device State

Ownership Rules	Configure mode	Execute/Idle Mode Unlocked	Execute/Idle Mode Locked
Config Owned (Config. OUNID = Owner UNID)	Type1 accepted by owner only	Type1 accepted by owner only	All Type 1 rejected
Config Un-owned (Config. OUNID = 0)	Type1 accepted from anyone	Type1 accepted from anyone	All type 1 rejected
SW Tool Owned (Config. OUNID = all 0xFF)	No Type 1 accepted	No Type 1 accepted	All type 1 rejected

#### 5-4.5.4 State Mapping of Safety Supervisor Object to Identity Object

Table 5-4.28 describes the mapping of Safety Supervisor states to states defined for the Identity object. This state mapping defines the interface between these three objects. The identity object state attribute (if supported) shall follow this table.



**Table 5-4.28 State Mapping of Safety Supervisor to Identity**

Safety Supervisor	Identity
Self-Testing	Device Self-Testing
Self-Test Exception	Major Unrecoverable
Idle	Standby
Configuring	Standby
Waiting for TUNID	Standby
Executing	Operational
Abort	Major Recoverable
Critical Fault	Major Unrecoverable

### 5-4.5.5 Safety Supervisor Event to Identity Object Event Mapping

Table 5-4.29 describes the mapping of defined Safety Supervisor events to equivalent events defined for the Identity object. This event mapping defines the interface between these objects. This mapping is provided for reference purposes.

**Table 5-4.29 Safety Supervisor Object Event Mapping**

Safety Supervisor State Events map to ->	Equivalent Identity Event
Power Applied	Power Applied
Self-Test Failed	Failed Tests
Self-Test Passed	Passed Tests
Propose TUNID	No equivalent event
Apply TUNID	No equivalent event
Exception Condition Cleared	Fault Corrected
Critical Fault	Major Unrecoverable Fault
Configure Request	Deactivated
Validate Configuration	No equivalent event
Apply Configuration	Activated
Type 1 SafetyOpen	No equivalent event
Reset Request	Reset
Abort (Internal)	Major Recoverable Fault
Recover Request	Fault Corrected
Perform Diagnostic Request	No equivalent event
Safety Connection Failed	Deactivated
Safety Connection Established	Activated
Safety Connection Deleted	No equivalent event
Mode Change	No equivalent event
Restore	No equivalent event
Lock/Unlock	No equivalent event



**5-4.5.6 Identity Object Event to Safety Supervisor Event Mapping**

Table 5-4.30 describes the mapping of defined Identity objects event conditions to equivalent events defined for the Identity object. This event mapping shows the equivalent events between these objects. This mapping is provided for reference purposes only; the safety supervisor object is the controlling object in safety devices and only events defined by the supervisor shall be implemented in safety devices.

**Table 5-4.30 Identity Object Event Mapping**

Identity Events map to ->	Safety Supervisor Events
Power Applied	Power Applied
Failed Tests	Self-Test Failed
Passed Tests	Self-Test Passed
Deactivated	Configure Request
Activated	Apply Configuration
Major Recoverable Fault	Abort (internal)
Major Unrecoverable Fault	Critical Fault
Minor Fault	No effect
Fault Corrected	Exception Condition Cleared
Reset	Service not supported error



## 5-5 Safety Validator Object

### Class Code: 3A<sub>hex</sub>

The Safety Validator Object is designed for use in Safety devices on all CIP Networks. The Safety Validator contains the information necessary to coordinate and maintain reliable safety connections between client and server safety applications. The primary role of the Safety Validator function is to act as a safety transport manager of multiple low-level CIP connections that together form a complete safety connection.

The device containing the Safety Validator Object can create several instances of the Safety Validator Object for each safety application connection point (e.g. application data assembly) supported by the device.

The Safety Validator Object supports two basic safety transport types: Single-Cast and Multi-Cast. Each safety transport type has a defined set of low-level CIP connections that must be allocated along with the Safety Validator object when a safety connection is instantiated by a Safety Open.

### 5-5.1 Revision History

Safety Validator Class Revision	Description
01	Initial Definition.

### 5-5.2 Class Attributes

Table 5-5.1 Safety Validator Class Attributes

Attr ID	Need in Implem	Access Rule	Name	Data Type	Description of Attribute	Semantics of Value
1 thru 7	Optional	Get			See the CIP Common Specification, Vol. 1, Chapter 4-9.1 for a description of the optional, reserved class attributes.	
8	Required	Get	Safety Connection Fault Count	UINT	Diagnostic Counter that is a running count of Safety Connection Faults	

#### 5-5.2.1 Safety Connection Fault Count

SRS75 This Safety Connection Fault Count attribute in the Safety Validator shall be a running count (with auto-rollover) of how many times any safety connection was faulted for any reason.



### 5-5.3 Instance Attributes

Most attribute values are derived, hard coded, or obtained from the dynamic connection establishment process. The only settable attribute value is the Max Data Age diagnostic.

**Table 5-5.2 Safety Validator Instance Attributes**

Attr ID	Need in Implementation	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
1	Required	Get		Safety Validator State	USINT	State of the Safety Connection	
2	Required	Get		Safety Validator Type	USINT	Safety Validator type used in this instance.	see semantics
3	Optional	Get		Ping Interval EPI Multiplier	UINT	Number that defines the Ping_Count_Int interval for a particular connection	16 to 1000
4	Optional	Get		Time Coord Msg Min Multiplier	STRUCT of		
				Time Coord Msg Min Multiplier array size	USINT	Size of array equals Max Consumer number for multi-cast producers and 1 for single-cast and multi-cast consumer	
				Time Coord Msg Min Multiplier	Array of. UINT	Minimum number of 128 uSec increments it could take for a Time Coordination Message to traverse from the consumer to the producer	0 - 7813



Attr ID	Need in Implementation	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
5	Optional	Get		Network Time Expectation Multiplier	STRUCT of:		
				Network Time Expectation Multiplier array size	USINT	Size of array equals Max Consumer number for multi-cast producers and 1 for single-cast and multi-cast consumer	
				Network Time Expectation Multiplier	Array of. UINT	Maximum number of 128 uSec increments that a consumer should allow the age of the safety data to reach	0 - 45313
6	Optional	Get		Timeout Multiplier	STRUCT of:		
				Timeout Multiplier array size	USINT	Size of array equals Max Consumer number for multi cast producers and 1 for single-cast and multi-cast consumer	
				Timeout Multiplier	Array of. USINT	Determines the number of messages that may be lost before declaring a connection error	BaseFormat1-4 Extended Format 1-255
7	Optional	Get		Max Consumer Number	USINT	Maximum number of consumers allowed for the connection	1 (Single-Cast), 2-15 (Multi-cast)



Attr ID	Need in Implementation	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
8	Optional <sup>1</sup>	Get		Data Connection Instance	UINT	Connection Instance ID of the Safety Data Connection	
9	Optional <sup>1</sup>	Get		Coordination Connection Instance	STRUCT of:	Connection Instance IDs of the Safety Coordination Connection	1-max number of connections in the device
				Coordination Connection Instance array size	USINT	Size of array equals Max Consumer number for multi-cast producers and 1 for single-cast and multi-cast consumer	
				Coordination Connection Instance	Array of: UINT		
10	Optional <sup>1</sup>	Get		Correction Connection Instance	UINT	Connection Instance ID of the Time Correction Connection	0 (when not used) 1-max number of connections in the device
11	Optional <sup>1</sup>	Get		CCO Binding	UINT	Instance Id of a CCO which holds the connection establishment settings for this connection. Only used by safety connection originators	0 (when not used or no binding present)
12	Required	Set		Max Data Age	UINT	Diagnostic which holds the largest Data Age detected in 128 $\mu$ S increments. Attribute only used for Safety Consumers	
13	Required	Get		Application Data Path	EPATH	Points to the application data attached to this safety connection	



Attr ID	Need in Implementation	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
14	Optional <sup>1</sup>	Get	V	Error Code	UINT	Reason for error within this instance.	0 if no error, else implementation specific error code.
15	Conditional <sup>2</sup>	Get	V	Producer/Consumer Fault Counters	STRUCT of.		
				Producer/Consumer Counter Array Size	USINT	Size of array equals Max Consumer number for multi-cast producers and 1 and 1 for single-cast and multi-cast consumer	
				Producer/Consumer Fault Counter	Array of USINT	Number of Faults detected this hour	See semantics

1. It is optional that these parameters be made publicly visible. However, all Validator instances shall have internal parameters which represent these variables.

2. If the device supports the EF format, this attribute is required

### 5-5.3.1 Safety Validator State

This attribute defines the current state of the Safety Validator Object instance. Transitions in this state attribute reflect the current state of the object (defined below). This attribute can be used to determine the condition of established safety connections.

**Table 5-5.3 Safety Validator State Assignments**

Value	State Name	Description
0	Unallocated	In this state the Safety Validator object has either not been allocated to a connection or has been closed.
1	Initializing	In this state the Safety Validator object is in the process of exchanging time coordination information across the connection. The “safety” connection is not yet fully established
2	Established	In this state the Safety Validator instance is fully established and producing/consuming safety data on behalf of the safety application
3	Connection_Failed	This state is entered when <b>all</b> connections associated with this validator have failed for any reason. In producers, all multi-cast consumer connections must have failed. As long as any consumer is still operating, the producer would remain in the Established state.
4-255	Reserved	

### 5-5.3.2 Safety Validator Type

This attribute defines the type of Safety Validator transport that is being used. Targets shall derive this value from the Safety Connection parameters in the Safety Open.

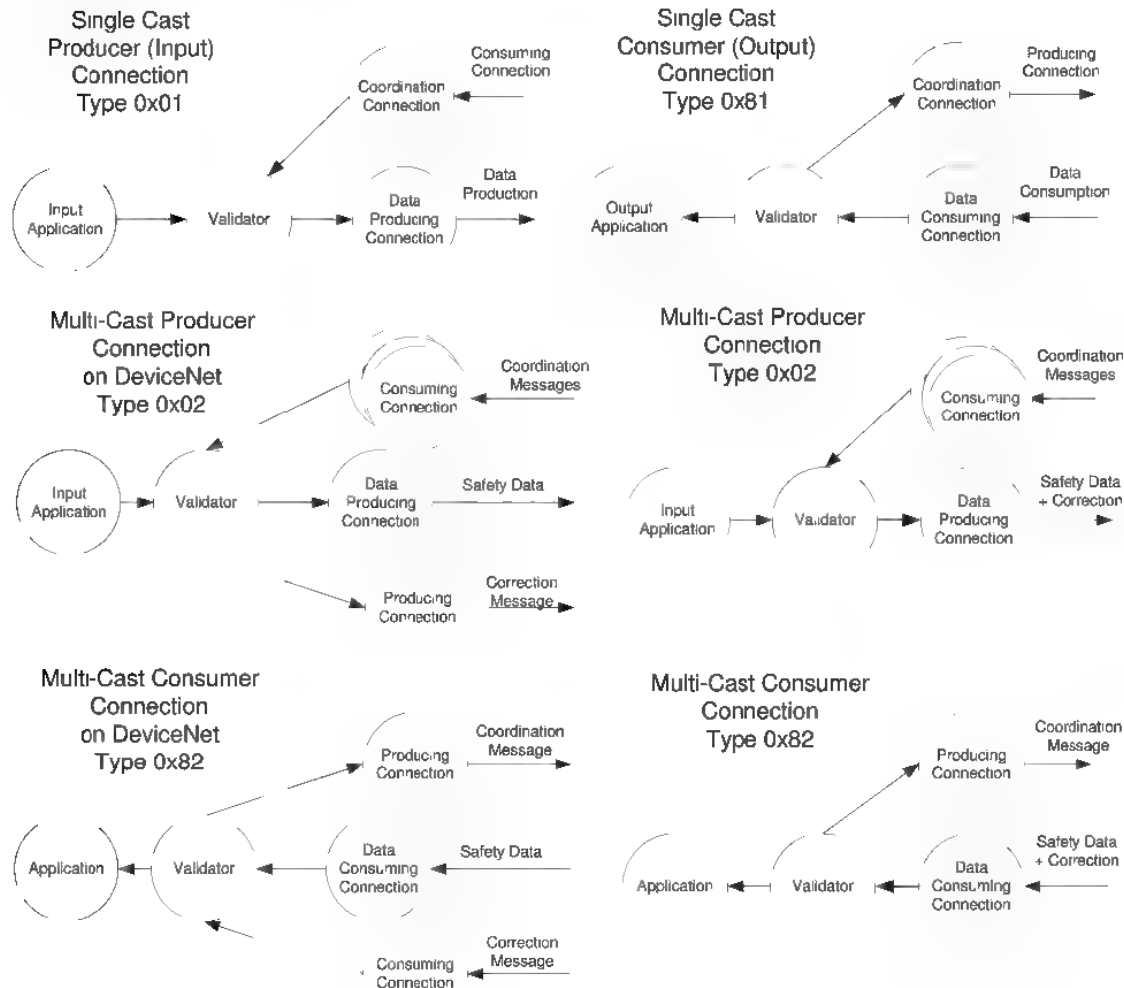


Table 5-5.4 Safety Validator Type, Bit field assignments

Bit 7	Bits 0 – 6
Producer(client) = 0/ Consumer(server) – 1	Safety Connection Type
	0 – unallocated
	1 = Single-cast
	2 = Multi-cast
	3-127 = Reserved

The transports for each Safety Validator Type are depicted below. This attribute can be used to query a device to determine the type of connection that has been established.

Figure 5-5.1 Safety Connection Types



### 5-5.3.3 Ping Interval EPI Multiplier

This attribute is used as part of the Safety Validator function. This parameter is obtained from the Safety Segment Parameters in the SafetyOpen. The rules Multi-cast producers should follow for evaluating this parameter from multiple consumers is shown in Table 5-5.5.



**5-5.3.4 Time Coord Msg Min Multiplier**

This attribute is used as part of the Safety Validator function. This parameter is obtained from the Safety Segment Parameters in the SafetyOpen. The rules Multi-cast producers should follow for evaluating this parameter from multiple consumers is shown in Table 5-5.5.

**Table 5-5.5 Multi-Cast Producer SafetyOpen Parameter Evaluation Rules**

Safety Parameter	Producer Evaluation Rule	Error Response (if appropriate)
Ping Interval EPI Multiplier	Must match existing P.I.E.M. (1 <sup>st</sup> consumer sets it)	Invalid Parameter
Time_Coord Msg Min Multiplier	Accept any (all can be different)	NA
Network Time Expectation Multiplier	Accept any (all can be different)	NA
Timeout Multiplier	Accept any (all can be different)	NA
Max Consumer Number	Must match existing Max_Consumer_Number (1 <sup>st</sup> consumer sets it)	Invalid Parameter

**5-5.3.5 Network Time Expectation Multiplier**

This attribute defines the connection reaction time associated with this connection. This parameter is obtained from the Safety Segment Parameters in the SafetyOpen. The rules Multi-cast producers should follow for evaluating this parameter from multiple consumers is shown in Table 5-5.5.

**5-5.3.6 Timeout Multiplier**

This attribute defines the tolerance the Safety Validator will have for lost messages before faulting a connection and taking a safety action. This parameter is obtained from the Safety Segment Parameters in the SafetyOpen. The rules Multi-cast producers should follow for evaluating this parameter from multiple consumers is shown in Table 5-5.5.

**5-5.3.7 Max Consumer Number**

This attribute defines the maximum number of consumers that are allowed to connect to a multi-cast producer (Type 82). Both Multi-cast Consumers and Producers must know this value to function properly. This parameter is obtained from the Safety Segment Parameters in the SafetyOpen. The rules Multi-cast producers should follow for evaluating this parameter from multiple consumers is shown in Table 5-5.5.

**5-5.3.8 Data Connection Instance**

This attribute identifies the instance of the connection object responsible for the Safety Data connection. This attribute is assigned as part of the connection establishment process.



#### 5-5.3.9 Coordination Connection Instance

This attribute identifies the instance of the connection object responsible for the Time Coordination connection. This attribute is assigned as part of the connection establishment process.

#### 5-5.3.10 Correction Connection Instance

This attribute identifies the instance of the connection object responsible for the Time Correction connection if used. This attribute is assigned as part of the connection establishment process.

#### 5-5.3.11 CCO Binding

This attribute identifies the Connection Configuration instance that was used to set up this safety connection. This binding is required to notify the CCO function when the connection is closed so that it can attempt to re-establish the connection. This attribute is only used in connection originators. This attribute may either be assigned as part of the connection establishment process or hard coded at design time.

#### 5-5.3.12 Max Data Age

The data age of received packet data is defined as the difference between the consumer's 128uS clock and received Time Stamp **at the time the packet is received**. The Max Data Age attribute is the maximum value this age has reached while the connection is/was established. The value can be used by diagnostic tools to fine tune the Network Time Expectation Multiplier.

SRS78 The Max Data Age attribute shall always reflect the largest value the connection's data age reaches without exceeding the expectation multiplier.

SRS79 If a safety connection is ever faulted and closed, the Max Data Age attribute shall be re-initialized to 0 when the connection is re-established.

#### 5-5.3.13 Producer/Consumer Fault Counter

For connections which are using the ExtendedFormat, this attribute reflects the internal Producer or Consumer Fault Counters defined in Chapter 2.

FRS375 When the connection is a EF Consuming connection, the Producer/Consumer Fault Counter attribute shall reflect the Consumer\_Fault\_Counters. When the connection is a EF Producing connection, the attribute shall reflect the Producer\_Fault\_Counter.

The attribute is used in the conformance tests which verify proper operation of the EF fault detection behavior.



**5-5.4 Class Services****Table 5-5.6 Safety Validator Class Services**

Service Code	Need in Implementation	Name	Description of Service
0E <sub>hex</sub>	Required	Get_Attribute_Single	Used to read a Safety Validator Object Class attribute value.
4B <sub>hex</sub>	Optional	Reset all error counters	Used to reset class attribute 8 in all instances

**5-5.5 Instance Services****Table 5-5.7 Safety Validator Instance Services**

Service Code	Need in Implementation	Name	Description of Service
01 <sub>hex</sub>	Optional	Get_Attribute_All	Returns the values of all instance attributes.
0E <sub>hex</sub>	Required	Get_Attribute_Single	Used to read a Safety Validator Object attribute value
10 <sub>hex</sub>	Required	Set_Attribute_Single	Used to reset the Max Data Age Attribute
4B <sub>hex</sub>	Optional	Reset error code	Used to reset instance attribute 14

**5-5.5.1 Get\_Attributes\_All Response**

For required attributes only.

**Table 5-5.8 Safety Validator Get\_Attributes\_All Service Data**

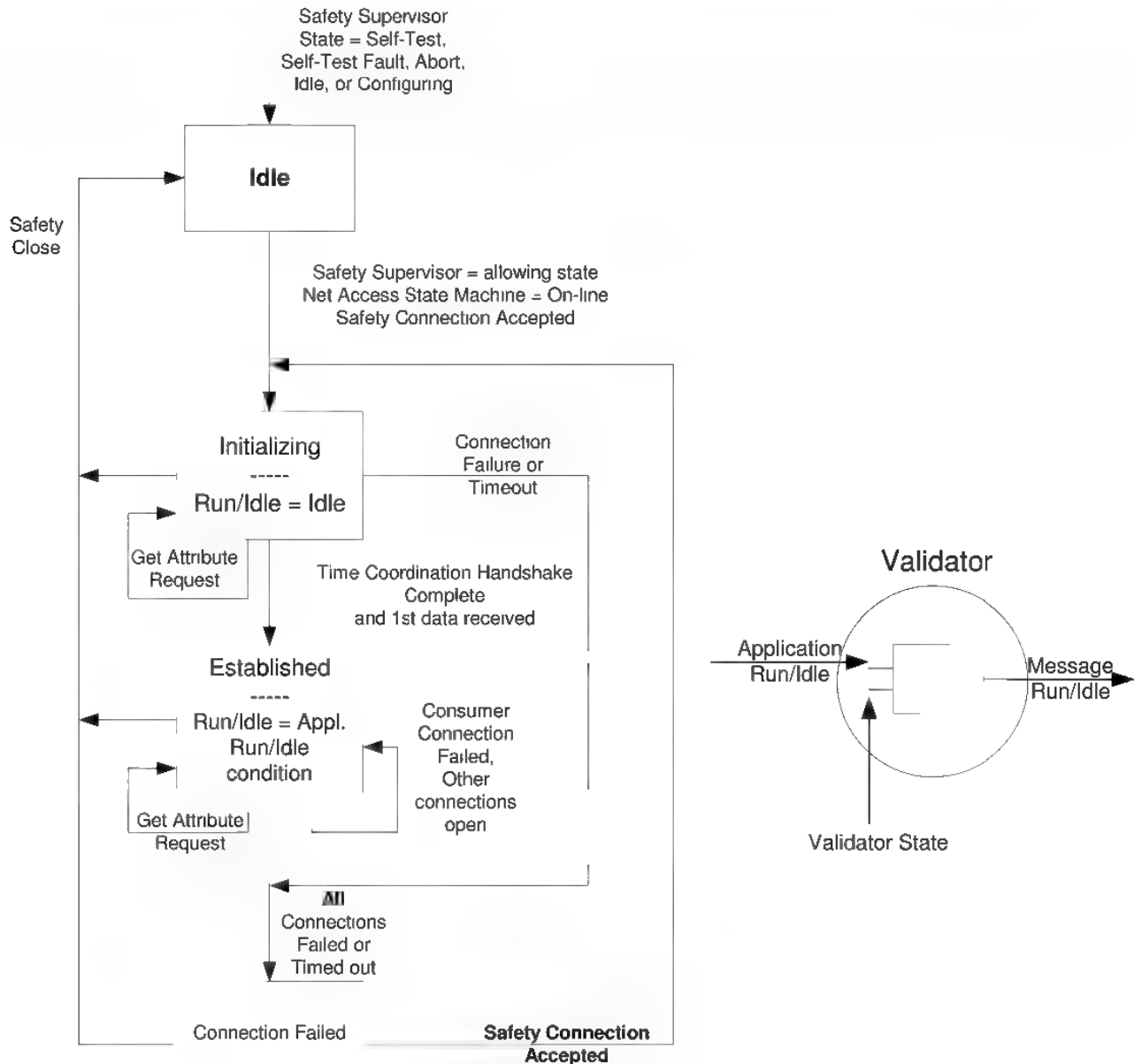
Attribute ID	Byte Number	Name	Data Type
1	0	Safety Validator State	USINT
2	1	Safety Validator Type	USINT
3	2,3	Ping Interval EPI Multiplier	UINT
4	variable	Time Coord Msg Min Multiplier	STRUCT
5	variable	Network Time Expectation	STRUCT
6	variable	Timeout Multiplier	STRUCT
7	variable	Max Consumer Number	USINT
8	variable	Data connection Instance	UINT = 0 if not supported
9	variable	Coordination Connection Instance	UINT = 0 if not supported
10	variable	Correction Connection Instance	STRUCT
11	variable	CCO Binding	UINT = 0 if not supported
12	variable	Max Data Age	UINT
13	variable	Application Path	EPATH
14		Error Code	UINT

**5-5.6 Object Behavior**

The behavior of the Safety Validator Object is illustrated in Figure 5-5.2 and Table 5-5.9 below.



Figure 5-5.2 Safety Validator State Transition Diagram



## 5-5.6.1 State Transition Diagram

### 5-5.6.1.1 IDLE

The Idle state is initial state entered after Power-up/Self-test. This state is the normal state when no connections are open or failed.



### 5-5.6.1.2 Initializing

For multi-cast producers, this state exists for each consumer connection handled by the producer. For single-cast producers and all consumers, this state relates to the entire Safety Validator connection.

### 5-5.6.1.3 Established

For multi-cast producers, this state exists for each consumer connection handled by the producer. For single-cast producers and all consumers, this state relates to the entire Safety Validator connection.

### 5-5.6.1.4 Connection\_Failed

This state is entered when all connections associated with the Safety Validator have failed. It can remain in this state until either a Safety Close is received, or the connection is re-established from a matching originator (i.e. OUNID and Application Path match a connection resource).

## 5-5.6.2 State Event Matrix

Table 5-5.9 Safety Validator State Event Matrix

Event	State			
	Idle	Initializing	Executing	Connection Failed
Power Applied and Self-Test Complete	Initial State	--	--	
Safety Supervisor = allowing state, Net Access State Machine = On-line, New Safety Connection Requested	Validate Request & Transition to Initializing	If multi-cast producer, Initialize Safety Connection	If Multi-cast producer, Initialize Safety Connection	Match OUNID & Application Path, If match, Transition to Initializing
Time Coordination Handshake Complete	--	Transition to Executing	If multi-cast producer, New Consumer On-line	--
Consumer fails in Multi-cast producer while other consumers still on-line	--	Flash NET status LED Red	Flash NET status LED Red	--
All connections Failed	--	Transition to Connection Failed	Transition to Connection Failed	Flash NET status LED Red
Matching Consumer Connection Request (i.e. OUNID and Application path match a failed consumer connection)	--	Transition Consumer to Initializing	Transition Consumer to Initializing	Transition to Initializing
Safety Close Received	Success Reply, Stay in Idle	Success Reply, Transition to Idle	Success Reply, Transition to Idle	Success Reply, Transition to Idle



**Table 5-5.10 State mapping between Safety Supervisor and Safety Validator objects**

<b>Safety Supervisor</b>	<b>Safety Validator</b>
Self-Testing	Idle
Self-Test Exception	Idle
Waiting for TUNID	Idle
Idle	Idle**, Initializing Executing, or Connection_Failed
Configuring	Idle
Executing	Idle**, Initializing or Executing, or Connection_Failed
Abort	Idle
Critical Fault	Idle

\*\*SRS80 A Safety Validator shall only accept connections when the Safety Supervisor is in the state that can accept connections which is device dependent (i.e. Executing only, or Execute/Idle) . A Safety Validator instance could be Idle while another Safety Validator is still active.



## 5-6 Connection Configuration Object

### Class Code: F3<sub>hex</sub>

This section defines the extensions to the Connection Configuration Object, Class 0xF3 for CIP Safety. (Also see CIP Common, Chapter 5 for the complete object definition.)

### 5-6.1 Class Attribute Extensions

The following class attributes are defined for use by CIP Safety devices.

**Table 5-6.1 Connection Configuration Object Class Attribute Extensions**

Attribute ID	Need in Implem	Access Rule	Name	Data Type	Description of Attribute
9	Conditional <sup>1</sup>	Set	Edit Signature	UDINT	Created and used by configuration software to detect modifications to the instance attribute values

1. This attribute is only required in devices not using the Safety SNCT interface.

### 5-6.2 Instance Attributes Additions and Extensions

The following instance attributes and changes are defined for use by CIP Safety devices.

**Table 5-6.2 Connection Configuration Object Instance Attribute Additions/Extensions**

Attr ID	Need in Implementation	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
9	Required	Set	NV	Data Map	STRUCT of		
				format number	UINT	This number determines the map format used.	2-99 Reserved 100 – 199 Vendor Specific All other values are Reserved.
				Data_Map data size	UINT	Size, in bytes, of Map information	
				Data_Map_data	Array of octet	Data Map format data based on Format number.	



**Connection Configuration Object, Class Code: F3<sub>Hex</sub>**

Attr ID	Need in Implementation	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
10	Optional	Set	NV	<b>Target Config Data</b>	STRUCT of		
				config_data_size	UINT	Length of config_data in bytes.	
				config_data	Array of octet	<b>Target Config</b> data	
11	Optional	Set	NV	Proxy Device ID	STRUCT of		
				vendor_id	UINT	Vendor ID	
				product_type	UINT	Device Type	
				product_code	UINT	Product Code	
				major_rev	USINT	Major revision	
				minor_rev	USINT	Minor revision	
12	Conditional <sup>3</sup>	Set	NV	Connection Disable	BOOL		<p>0 – connection is enabled.</p> <p>1 – connection is disabled.</p> <p>When an Open service is received this value shall be set to 0. When a Close or Stop service is received this value shall be set to 1.</p>
13	Conditional <sup>3</sup>	Set	NV	Safety Parameters	STRUCT of		
				reserved	USINT	Reserved	Shall be zero
				reserved	USINT	Reserved	Shall be zero
				Configuration CRC	UDINT	CRC32 over the target device configuration data in Attribute 7 & 10	This value calculated by the configuration software on the data in Attr. 7 & 10 after the configuration is applied
				Configuration Signature	DATE AND TIME		
				Time Correction EPI	UDINT		
				Time Correction Connection Parameters	WORD		
				Target UNID	10:octets		UNID of target device



**Connection Configuration Object, Class Code: F3<sub>Hex</sub>**

Attr ID	Need in Implementation	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
				Reserved space for Originator UNID	10:octets	All zeros	Allows FW to easily stuff OUNID into safety segment
				Ping Int EPI Mult	UINT		
				Time Coord Msg Min Mult	UINT		
				Net Time Exp Mult	UINT		
				Timeout Multiplier	USINT		
				Max Consumer Number	USINT		
14	Conditional <sup>3</sup>	Get	NV	Connection Parameter CRC	UDINT	CRC-32 over the connection parameters used in a Safety Open	This value is calculated by this device only once when the configuration is applied. It must be readable by configuration software
15	Conditional <sup>3</sup>	Set	NV	Configuration Instance	UINT	Which object has configuration for this node	= 0 when no object assigned
16	Conditional <sup>3</sup>	Set	NV	Id Allocation	BOOL	When cleared, the originator will allocate the network Ids for its produced connections	0: allocate the Id for produced connections (default) 1; ask target for produced Ids
17	Conditional <sup>3</sup>	Get		Target Connection Serial Number	UINT	The instance # of the Target's Safety Validator.  Target Connection Serial Number returned during connection establishment. Read by diagnostic tools	0 when connection is not established
20	Conditional <sup>4</sup>	Set	NV	Format Type	USINT	Defines which format to use in connecting to a target	0 = Auto (default) 1 = "Base" 2 = "Extended"



**Volume 5: CIP Safety, Chapter 5: Object Library**  
**Connection Configuration Object, Class Code: F3<sub>Hex</sub>**

Attr ID	Need in Implementation	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
21	Conditional <sup>4</sup>	Get	NV	Format Status	USINT	Indicates which format is currently in use when the Format Type is Auto (or if no agreement on a format could be obtained)	0 = Auto 1 = "Base" 2 = "Extended" 0xFF = "conflict detected"
22	Conditional <sup>4</sup>	Set	NV	Max Fault Number	UINT	Used by both producers and consumers to determine how many erroneous packets can be dropped before a connection must be closed.	Default = 5. A value of 0 indicates connections must be closed on any detected error.

<sup>3</sup>This attribute is required for a device which supports safety connections; otherwise this attribute is optional

<sup>4</sup>This attribute is required for a device which supports the Extended format; otherwise this attribute is optional

## 5-6.2.1 Instance Attribute Semantics Extensions or Restrictions for Safety

### 5-6.2.1.1 Connection Flags – (Attribute 2)

**Table 5-6.3 Connection Flags Attribute Definition**

Bit	Meaning
0	Connection Direction 0 = Originator 1 = Target
1 - 3	O->T Real time transfer format 0 = Use 32-bit Run/Program header to indicate idle mode. 1 = Use zero data length packet to indicate idle mode. 2 = None. Connection is pure data and is modeless. 3 = Heartbeat. 4 = Reserved for future use 5 = Safety Format Other values are reserved for future use.
4 - 6	T->O Real time transfer format 0 = Use 32-bit Run/Program header to indicate idle mode. 1 = Use zero data length packet to indicate idle mode. 2 = None. Connection is pure data and is modeless. 3 = Heartbeat. 4 = Reserved for future use 5 = Safety Format Other values are reserved for future use.
7 - 15	Reserved



**Connection Flag usage for safety**

Connection Flags are defined as follows:

- Bit 0: Connection Direction.  
Always 0 = Originator of connection
- Bit 1-3: O⇒T Real time transfer format  
Always 101 = Safety Format
- Bit 4-6: T⇒O Real time transfer format  
Always 101 = Safety Format
- Bit 7-15: Not used.  
Always 0.
- Value:** 0x005A = Originator  
0x005B = Target

**5-6.2.1.2 CS Data Index Number – (Attribute 4)**

**CS Data Index usage for safety**

Not Used. Must be 0xFFFFFFFF

**5-6.2.1.3 Connection Timeout Multiplier – (part of Attribute 5)**

**Connection Timeout Multiplier usage for safety**

It is used by the producer and consumer to timeout any of the 3 standard connections. It is used by the consumer of the connection to determine if the connection should timeout. The timeout value for the connection is defined as:

Connection RPI \* (CTM + 1) \* 4

**Range:** 0 to 7

0 = (x4), 1 = (x8), 2 = (x12), 3 = (x16), 4 = (x20), 5 = (x24), 6 = (x28), 7 = (x32)

**Value:** Always 1 (x8)

**5-6.2.1.4 Transport Class and Trigger – (part of Attribute 5)**

**Transport Class and Trigger usage for safety**

This attribute defines the transport class and the trigger as defined below:

- Bit 0-3: Transport Class  
Always 0000 – Class 0
- Bit 4-6: Production Trigger  
Always 010 (Application)
- Bit 7: Direction  
0 = Client (T⇒O used for data, i.e. input device)  
1 = Server (O⇒T used for data, i.e. output device)
- Value:** 0x20 (0010 0000) for an Input Connection  
0xA0 (1010 0000) for an Output Connection



#### **5-6.2.1.5      O=>T RPI – (part of Attribute 5)**

##### **O=>T RPI usage for safety**

If direction is 0 (Client / T=>O), then this is the Time Coordination RPI and the T=>O RPI is the Data Production RPI.

If direction is 1 (Server / O=>T), then this is the Data Production RPI and the T=>O RPI is the Time Coordination RPI.

##### **Time Coordination RPI**

**Range:**     100 to 100,000,000 us

Ping Interval EPI Multiplier \* Data Production RPI

##### **Data Production RPI**

**Range:**     100 to 1,000,000 us

#### **5-6.2.1.6      O=>T Connection Parameters – (part of Attribute 5)**

##### **O=>T Connection Parameters usage for safety**

Defined as follows:

Bits 0-8:            Connection size (in bytes).

If the direction is 1 (Server / O=>T), it is the size of the Safety Message:

If Data Size is 1-2 bytes: Data Size = application data size + 6 bytes

If Data Size is 3-250 bytes: Data Size = (application data size \* 2) + 8 bytes

If the direction is 0 (Client / T=>O), it is the size of the Time Coordination Message (6 bytes).

**Range:** 7-510

Bit 9:        Fixed / Variable  
              Always 0 (Fixed)

Bit 10 11: Priority  
              Always 01 (High)

Bit 12:       Reserved  
              Always 0

Bit 13-14: Connection Type  
              Always 10 (Point to Point)

Bit 15:       Redundant Owner  
              Always 0 (Safety doesn't support Redundancy)



#### **5-6.2.1.7      T=>O RPI – (part of Attribute 5)**

##### **T=>O RPI usage for safety**

If direction is 0 (Client / T=>O), then this is the Data Production RPI and the O=>T RPI is the Time Coordination RPI.

If direction is 1 (Server / O=>T), then this is the Time Coordination RPI and the O=>T RPI is the Data Production RPI.

**Value:**      Refer to O=>T RPI

#### **5-6.2.1.8      T=>O Connection Parameters – (part of Attribute 5)**

##### **T=>O Connection Parameters usage for safety**

Defined as follows:

Bits 0-8:      Connection size (in bytes).

If the direction is 0 (Client / T=>O), it is the size of the Safety Message.

For Single-cast, see O=>T Connection Parameters.

For Multi-cast with a Data Size of 1-2: Data Size = application data size + 12

For Multi-cast with a Data Size of 3-247: Data Size = (application data size \* 2) + 14

Multi-cast calculations shall include the Time Correction Section. This is required for all multi-cast connections to ensure the proper calculation of the CPCRC in the SafetyOpen. The value is fixed up by the Target Device.

If the direction is 1 (Server / O=>T ), it is the size of the Time Coordination Message (6 bytes).

**Range:** 508

Bit 9: Fixed / Variable

Always 0 (Fixed)

Bit 10-11:      Priority

Always 01 (High)

Bit 12:      Reserved

Always 0

Bit 13-14:      Connection Type

01 (Multi-cast)

10 (Point to Point)

Default: 10 (Point to Point)

Bit 15:      Redundant Owner

Always 0 (Safety doesn't support Redundancy)



#### **5-6.2.1.9 Connection Path – (Attribute 6)**

Connection Path Size - Number of 16-bit words stored in Connection Path. This path shall be adjusted to remove the size of the bridge path elements when the CPCRC is calculated (i.e. the size used in the calculation must be the value the target will see after the bridge(s) have stripped away their segments and appropriately adjusted this size).

##### **Connection Path usage for safety**

The connection paths associated with the connection in the following order: (1) Bridge Path, (2) Configuration Path, (3) Consumption Path, and (4) Production Path.

The bridge path (1) shall be separated by the originator from the application path 2, 3, & 4 at the time the SafetyOpen is built and when the CPCRC is calculated.

##### **Bridge Path**

Valid path constructed for the route between the originator and target.

##### **Configuration Path**

Valid path if configuration data is present in Attributes 7 or 8, or Null path otherwise. The path in all cases comes from the Target Applet (i.e. EDS File).

##### **Target Consumption Path**

Valid path if direction 1 (Server / O=>T) or Null Path otherwise. The path in all cases comes from the Target Applet (i.e. EDS File).

##### **Target Production Path**

Valid path if direction is 0 (Client / T=>O) or Null Path otherwise. The path in all cases comes from the Target Applet (i.e. EDS File).

#### **5-6.2.1.10 Proxy Config Data Size – (part of Attribute 7)**

Usage is optional for Safety, zero if not used

#### **5-6.2.1.11 Proxy Config Data – (part of Attribute 7)**

Usage is optional for Safety and should be concatenated with the contents of Target Config data

#### **5-6.2.1.12 Target Config Data Size – (part of Attribute 10)**

##### **Target Config Data Size usage for safety**

Number of bytes stored in Target Config Data (below)

Range: 1 to <Max Allowable Configuration Size>

#### **5-6.2.1.13 Target Config Data – (part of Attribute 10)**

##### **Target Config Data usage for safety**



Used to store the target device's configuration data

#### 5-6.2.1.14 Data Map – (Attribute 9)

This attribute name has been changed from “Implementation Defined” to “Data Map” to reflect its primary use; to map the application data to this connection. What follows is a set of map formats that are currently defined in the standard range. The vendor specific range is still available to be used for special map formats.

**Table 5-6.4 Data Map Formats**

Format Number	Format Name	Data Map Data Size (in octets)	Data Map Data	Comment
0	Open Scanner Format	4 octets	See Table 5-6.5	Simple format which maps data in word granularity to input and output image tables
1	Byte-Index Scanner Format	4 octets	See Table 5-6.6	Simple format which maps data in byte granularity to input and output image tables
2-99	Reserved			
100-199	Vendor specific			
200 - 65535	Reserved			

##### 5-6.2.1.14.1 Format 0 Usage for Safety Scanners

Contains the Map Data structure using the format described in Map Format Type.

**Table 5-6.5 Map Data for Format 0**

Name	Data Type	Description
Originator to Target Data image offset	UINT	The offset, in 16 bit units, in the originator to target image table
Target to Originator Data image offset	UINT	The offset, in 16 bit units, in the target to originator image table

**Values:**

**O⇒T Data Image Offset**

If the direction is 0 (Client / T⇒O): Always 0.

If the direction is 1 (Server / O⇒T): Word offset into the Safety Output Data Table.

**T⇒O Data Image Offset**

If the direction is 0 (Client / T⇒O): Word offset into the Safety Input Data Table.

If the direction is 1 (Server / O⇒T): Always 0.



#### **5-6.2.1.14.2 Format 1 Usage for Safety Scanners**

Contains the Map Data structure using the format described in Map Format Type.

**Table 5-6.6 Map Data for Format 1**

Name	Data Type	Description
Originator to Target Data image offset	UINT	The offset, in 8 bit units, in the originator to target image table
Target to Originator Data image offset	UINT	The offset, in 8 bit units, in the target to originator image table

Values:

**O⇒T Data Image Offset**

If the direction is 0 (Client / T⇒O): Always 0.

If the direction is 1 (Server / O⇒T): byte offset into the Safety Output Data Table.

**T⇒O Data Image Offset**

If the direction is 0 (Client / T⇒O): byte offset into the Safety Input Data Table.

If the direction is 1 (Server / O⇒T): Always 0.

#### **5-6.2.1.15 Proxy Device ID**

Proxy Device ID usage for safety: Never used

#### **5-6.2.1.16 Connection Disable (Attribute 12)**

FRS356 When the Connection Disable attribute in the CCO object is TRUE, the connection shall remain disabled through power cycles. When the Connection Disable attribute in the CCO object is FALSE, the connection shall remain enabled through power cycles.

### **5-6.2.2 Special Safety Related Parameters – (Attribute 13)**

#### **5-6.2.2.1 Ping Interval EPI Multiplier**

This attribute is used as part of the safety Validator function. This parameter is part of the Safety Segment Parameters in the Safety Open. Ping\_Interval\_EPI\_Multiplier is the number that defines the Ping\_Count Interval for the safety connection.

SRS85 Devices implementing the safety extensions to the CCO object shall perform a range check on the Ping Interval EPI Multiplier during a Validate\_Configuration request

**Ping Interval EPI Multiplier usage**

Range: [Max\_Consumer\_Number\*Timeout\_Multiplier.PI +15] to 1000.

Default:

If the direction is 0 (Client / T⇒O): 100

If the direction is 1 (Server / O⇒T): 19



#### **5-6.2.2.2 Time Coord Msg Min Multiplier**

This attribute is used as part of the Safety Validator function. This parameter is obtained from the safety segment parameters in the SafetyOpen. Time\_Coord\_Msg\_Min\_Multiplier is the minimum number of 128 uS increments it could take for a Time Coordination Message to traverse from the consumer to the producer. The default is 0.

SRS86 Devices implementing the safety extensions to the CCO object shall perform a range check on the Time\_Coord\_Msg\_Min\_Multiplier during a Validate\_Configuration request.

##### **Time Coord Msg Min Multiplier usage**

Range: 0 to 7813, which equates to 0 to 1 sec.

Default: 2 for non-bridge; 3 or more when bridging is added.

#### **5-6.2.2.3 Network Time Expectation Multiplier**

This attribute is used as part of the Safety Validator function. This parameter is part of the safety segment parameters in the SafetyOpen. Network\_Time\_Expectation\_Multiplier is the maximum number of 128 uSec increments that a consumer should allow the age of the safety data to reach.

SRS87 Device implementing the safety extensions to the CCO object shall perform a range check on the Network\_Time\_Expectation\_Multiplier during a Validate\_Configuration request.

##### **Network Time Expectation Multiplier usage**

Range 0 to 45313, which equates to 1 to 5.8 sec.

Equation: the smaller of 45313 or

$$NTEM = [(Timeout\_Multiplier.PI + 2) * Data\_Production\_RPI]/128$$

#### **5-6.2.2.4 Timeout Multiplier**

This attribute is used as part of the Safety Validator function. This parameter is part of the safety segment parameters in the SafetyOpen.

SRS88 Device implementing the safety extensions to the CCO object shall perform a range check on the Timeout\_Multiplier during a Validate\_Configuration request.

##### **Timeout Multiplier usage**

Range BaseFormat: 1 to 4

Range ExtendedFormat: 1 to 255

Default: 2

#### **5-6.2.2.5 Max Consumer Number**

This attribute defines the maximum number of consumers that are allowed to connect to a multi-cast producer (Type 2). Both Multi-cast Consumers and Producers must know this value to function properly. This attribute is used as part of the safety Validator function. This parameter is part of the Safety Segment Parameters in the Safety Open.



SRS89 Device implementing the safety extensions to the CCO object shall perform a range check on the Max Consumer Number during a Validate\_Configuration request.

**Max Consumer Number usage**

Range: 1 to 15

Default: 15

Value: Comes from the Target Applet (i.e. EDS File)

### 5-6.2.2.6 Target Connection UNID

This attribute is the UNID of the target safety device for this connection. This parameter is part of the Safety Segment Parameters in the Safety Open and is checked by the target device to confirm the request didn't get misrouted. The only way to confirm this value is correct is through user testing of the connection.

**Target UNID format on DeviceNet**

Byte 0-5: Safety Network Number

Byte 6: DeviceNet MacID

Byte 7-9: Always 0 for DeviceNet

### 5-6.2.2.7 Safety Configuration CRC (SCCRC)

The SCCRC serves as a signature for the device configuration and can be used to confirm the integrity of the configuration over time. The configuration software shall calculate the SCCRC over the configuration data in the exact order it appears in the SafetyOpen message. This same calculation is done by target devices and compared to this value.

SRS90 The SCCRC shall be calculated over the configuration data by the device whenever a device configuration is validated (i.e. Validate\_Configuration Request to the safety supervisor or type 1 safety connection)

SRS91 The SCCRC shall be saved to NV storage along with the other NV attributes whenever a configuration is applied (i.e. apply request to the safety supervisor).

SRS92 The user shall be instructed in the safety manual to test safety connection configurations after they are applied in an originator to confirm the target connection is operating as intended.

**Safety Configuration CRC usage**

The target device's SCCRC.

**Value:** Based on the type of safety connection.

Type 1: SCCRC identifying configuration data in **Target Config Data**.

Type 2a: SCCRC identifying the expected target device configuration.

Type 2b: 0



#### **5-6.2.2.8 Configuration Signature (Time Stamp)**

The target device's SCTS.

##### **Configuration Time Stamp usage**

Value: Based on the type of safety connection.  
Type 1: SCTS identifying configuration data in **Target Config Data**.  
Type 2a: SCTS identifying the expected target device configuration.  
Type 2b: 0

#### **5-6.2.2.9 Time Correction EPI**

Defines the EPI for the Time Correction Message used in a multi-cast connection.

##### **Time Correction EPI usage**

Value: For a multi-cast connection, it is  
Ping Interval EPI Multiplier \* Data Production RPI  
Otherwise 0.

#### **5-6.2.2.10 Time Correction Connection Parameters**

Defines the Connection Parameters used in the Time Correction Message used in a multi-cast connection.

##### **Time Correction Connection Parameter usage**

Value: 0x0000 for a non Multi-cast connection.  
0x2406 for a Multi-cast connection, defined as follows:  
Bits 0-8: Size of the Time Correction Message which is 6 bytes.  
Bit 9: Fixed / Variable  
0 (Fixed)  
Bit 10-11: Priority  
01 (High)  
Bit 12: Reserved  
Always 0  
Bit 13-14: Connection Type  
01 (Multi-cast)  
Bit 15: Redundant Owner  
Always 0 (Safety doesn't support Redundancy)

#### **5-6.2.3 Connection Parameter CRC (CPCRC) – (Attribute 14)**

SRS93 The Connection Parameter CRC attribute shall be calculated by this originator device whenever the originator's configuration is validated (i.e. Validate\_Configuration request to the safety supervisor).

SRS94 The CPCRC shall be saved to NV storage along with the other NV attributes whenever a configuration is applied (i.e. apply request to the safety supervisor).



The user should be required to test this connection after it is applied to confirm the connection configuration. From that point on, the CPCRC serves as a signature for that connection configuration and can be used to confirm the integrity of the configuration over time. The calculation itself is done over the connection parameters in Safety Open in the exact order they appear in the message. This same calculation is done by target devices and compared to this value, so the exact order must be followed.

#### **5-6.2.4 Configuration Instance – (Attribute 15)**

This attribute provides the linkage to the instance of the object assigned to this connection. The attribute may be used when the target's configuration data cannot be sent via the SafetyOpen. A value of 0 (default) for this attribute indicates that no object is assigned.

#### **5-6.2.5 Id Allocation**

This Boolean attribute is used to give the user some flexibility in how originators allocate Ids on connection establishment. This is particularly important for DeviceNet Safety. The default behavior is for the originator to provide Ids for the connections in which it is the producer. If the user sets this attribute, the originator will instead ask the Target to provide an Id for these produced connections. A user may want to do this when the originator has a large number of connections and does not have enough to do the application. This attribute only applies to non-multi-cast productions (multi-cast productions must always be allocated by the producer). This function should be considered an advanced function and great care should be taken when presenting this feature to a user. The user must be advised it should only used in limited application scenarios where connection repeatability can be guaranteed.

#### **5-6.2.6 Format Type**

The following table defines the Format Type attribute meaning.

**Table 5-6.7 Meaning of the Format Type Attribute**

<b>Format Type Value Name</b>	<b>Meaning</b>
Auto	The Originator will determine which format to use at first connection establishment.
Base	The Originator will use the existing format at the next connection establishment.
Extended	The Originator will use the Extended format at the next connection establishment.

If the Format Type attribute is not set it will default to Auto the originator will determine which format to use. If the Format Type attribute is optionally configured to "Base", the originator will only use the "Base" format. If the Format Type attribute is optionally configured to "Extended", the originator will only use the "Extended" format.

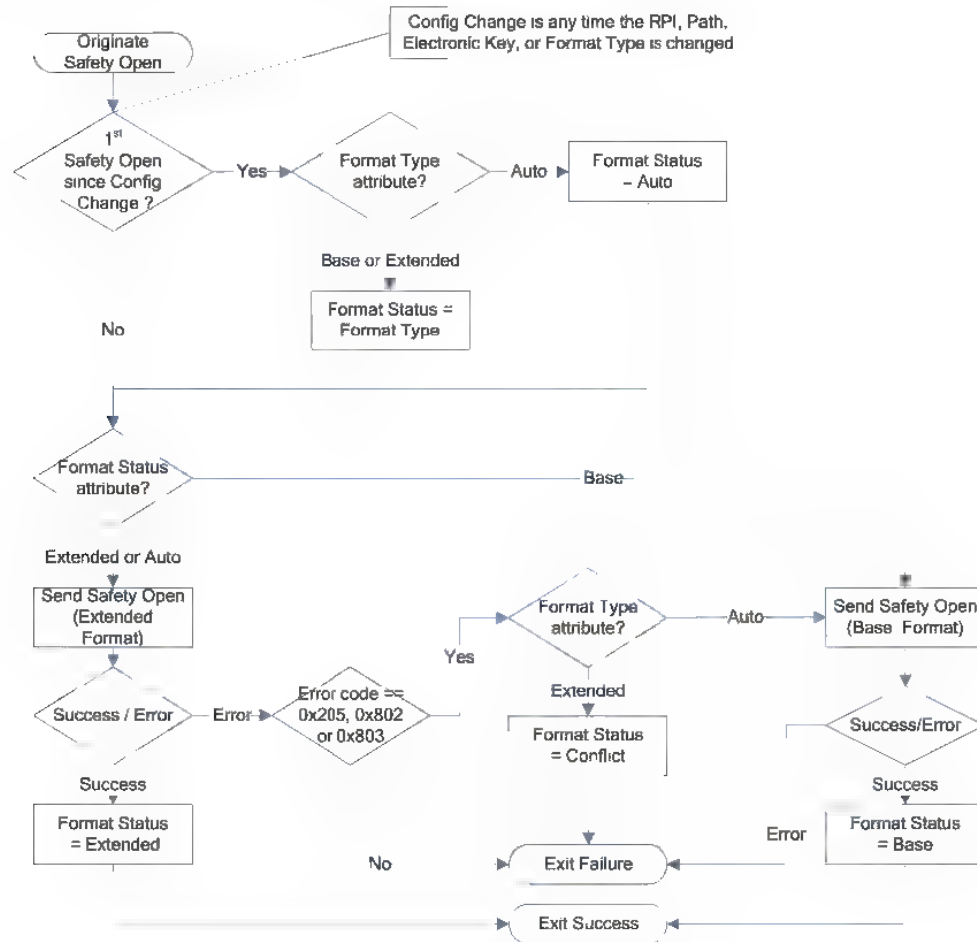
If the Format Type is Base or Extended and a Target responds with a "Invalid Safety Connection Format" Error (0x01, 0x803) or "Invalid Safety Connection Size" Error (0x01, 0x802), the Format Status attribute shall be set to 0xFF (Conflict Detected).



**Volume 5: CIP Safety, Chapter 5: Object Library**  
**Connection Configuration Object, Class Code: F3<sub>Hex</sub>**

The Format Type attribute can not be changed while the connection is active. If the format is set to Auto, the following logic shown in Figure 5-6.1 be used:

**Figure 5-6.1 Logic for Auto-detecting format type**





### 5-6.2.7 Format Status

The following table defines the Format Status attribute meaning.

**Table 5-6.8 Meaning of the Format Status Attribute**

Format Status Value Name	Meaning Originator	Meaning Target
Auto	The Originator has not yet determined which format to use.	NA
“Base”	If Format Type is Auto The Originator has determined target requires old format If Format type is ‘Base, The originator will only connect with old format If Format type is ‘Extended This condition should never occur	The target is using the “Base” format
“Extended”	If Format Type is Auto The Originator has determined target will accept new format If Format type is ‘Base, The condition should never occur If Format type is ‘Extended The originator will only connect with the new format	The target is using the “Extended”
‘Conflict Detected’	If Format type is Base or Extended and Segment type errors are received, there is setting conflict between Target and Originator. This status flags this condition	The target is receiving SafetyOpen requests for a format it doesn’t support



### 5-6.3 Object-Specific Services

The Connection Configuration Object provides the following Object Specific Services:

**Table 5-6.9 Connection Configuration Object, Object-specific Services**

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
4B <sub>hex</sub>	Conditional <sup>1</sup>	N/A	Kick-Timer	Kicks Edit Watchdog Timer
4C <sub>hex</sub>	Optional	Conditional <sup>2</sup>	Open	Opens connections
4D <sub>hex</sub>	Optional	Conditional <sup>2</sup>	Close	Closes connections
4E <sub>hex</sub>	Optional	Conditional <sup>2</sup>	Stop	Stops connections
4F <sub>hex</sub>	Conditional <sup>1</sup>	N/A	Change Start	Manages session editing
50 <sub>hex</sub>	Required	N/A	Get Status	Get status for multiple connections
51 <sub>hex</sub>	Conditional <sup>1</sup>	N/A	Change Complete	Completes session editing
52 <sub>hex</sub>	Conditional <sup>1</sup>	N/A	Audit Changes	Audits pending changes

1. Safety Devices shall not implement these services since the software configuration tool will coordinate configuration through the Safety Supervisor.
2. FRS357 Safety Devices shall support the Open, Close and Stop services. These services shall only be accepted when the device is unlocked. If the device is locked, it shall respond with a "Device state conflict" error.

### 5-6.4 Common Service Extensions for Safety

#### 5-6.4.1 Get Attribute All (Service Code 0x01)

The added attributes to the end of the Get\_Attribute\_All response is shown in the two tables below. All other parameters of this service are as defined in the standard CCO object in CIP Specification Vol.1, Chapter 5. Refer to CCO object in CIP Specification Vol.1, Chapter 5 for the position of the two parameter fields in the Get\_Attribute\_All service.



**Volume 5: CIP Safety, Chapter 5: Object Library**  
**Connection Configuration Object, Class Code: F3<sub>Hex</sub>**

**Table 5-6.10 Get\_Attributes\_All Response Service Data Safety Parameters**

Name	Data Type	Description
Safety Parameters	STRUCT of	
	USINT	Pad, set to zero
	USINT	Pad, set to zero
	UDINT	Configuration CRC (SCCRC
	DATE AND TIME	Configuration Time Stamp (SCTS)
	UDINT	Time Correction EPI
	WORD	Time Correction Connection Parameters
	10:octets	Target UNID
	10:octets	Pad, set to zero
	UINT	Ping Int EPI Mult
	UINT	Time Coord Msg Min Mult
	UINT	Net Time Exp Mult
	USINT	Timeout Multiplier
	USINT	Max Consumer Number
Connection Parameter CRC	UDINT	CRC-32 over the connection parameters used in a Safety Open
Configuration Instance	UINT	
ID Allocation Flag	BOOL	
Target Connection Serial Number	UINT	Safety Validator Instance returned from Target during connection establishment

**Table 5-6.11 Get\_Attributes\_All Response Service Data Additional Safety Parameters**

Name	Data Type	Description
Length	USINT	Length of additional safety parameters = 4
Format Type	USINT	
Format Status	USINT	
Max Fault Number	UINT	



#### **5-6.4.2 Set\_Attribute\_All (Service Code 0x02)**

The added attributes to the end of the Set\_Attribute\_All response shall be extended as shown in the two tables below. All other parameters of this service are as defined in the standard CCO object in CIP Specifications Vol.1, Chapter 5. Refer to CCO object in CIP Specification Vol.1, Chapter 5 for the exact position of the two parameter fields in the Set\_Attribute\_All service.

**Table 5-6.12 Set Attributes All Request Service Data (Added Attributes)**

Name	Data Type	Description
Safety Parameters	STRUCT of	
	USINT	Pad, set to zero
	USINT	Pad, set to zero
	UDINT	Configuration CRC (SCCRC) (value set = 0, device does this calc)
	DATE AND TIME	Configuration Time Stamp (SCTS) (if applicable, else = 0)
	UDINT	Time Correction EPI
	WORD	Time Correction Connection Parameters
	10:octets	Target UNID
	10:octets	Pad, set to zero
	UINT	Ping Int EPI Mult
	UINT	Time Coord Msg Min Mult
	UINT	Net Time Exp Mult
	USINT	Timeout Multiplier
	USINT	Max Consumer Number
Configuration Instance	UINT	
ID Allocation Flag	BOOL	

**Table 5-6.13 Set Attributes All Request Service Data Additional Parameters**

Name	Data Type	Description
Length	USINT	Length of additional safety parameters = 3
Format Type	USINT	
Max Fault Number	UINT	

#### **5-6.4.3 Restore (Service Code 0x15)**

Safety devices shall not implement the Restore service at either class or instance level since the configuration tool will coordinate configuration through the Safety Supervisor.

#### **5-6.5 Object Behavior**

The Connection Configuration object behavior for Safety Devices shall be controlled by the Safety Supervisor and follow the behavior shown in Table 5 6.14, Figure 5 6.2, and Figure 5 6.3.

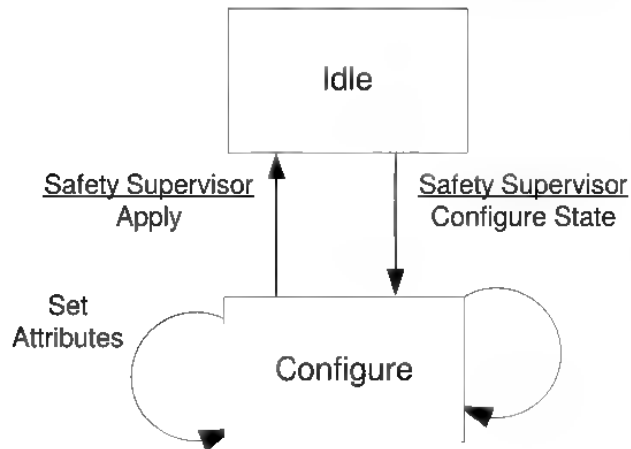
State mapping between Safety Supervisor and Connection Configuration objects



**Table 5-6.14 State Mapping between Safety Supervisor and the CCO objects**

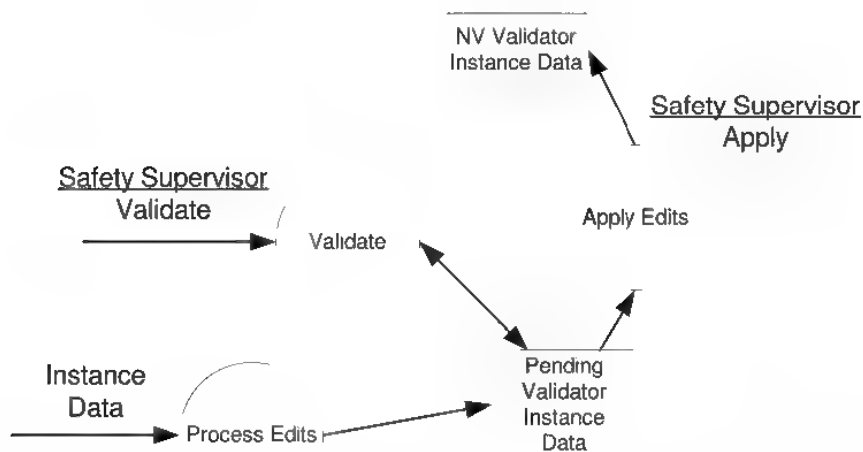
Safety Supervisor	Connection Configuration
Self-Testing	Edits Not allowed
Self-Test Exception	Edits Not allowed
Idle	Edits Not allowed
Configuring	Edits allowed
Executing	Edits Not allowed
Abort	Edits Not allowed
Critical Fault	Edits Not allowed

**Figure 5-6.2 Connection Configuration Object State Diagram**



The Data flow for the Connection Configuration object while in the Configure State is shown in Figure 5-6.3

**Figure 5-6.3 Connection Configuration Object Data Flow**





## 5-7 Safety Discrete Input Point (SDIP)

### Class Code: 3D<sub>hex</sub>

The Safety Discrete Input Point Object models discrete safety inputs in a product. This object can be used in any device that contains discrete safety inputs. Note that the term “input” is defined from the network’s point of view. An input will produce data on the network.

The Safety Discrete Input Point interface is to real input points such as a switch or screw terminal. The input is sampled and evaluated, the evaluated result is stored in this object’s Safety Input Logical Value attribute.

### 5-7.1 Revision History

Revision	
01	Initial definition at First Release of the Object

### 5-7.2 Class Attributes

Table 5-7.1 Safety Discrete Input Point Class Attributes

Attr ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
1 thru 7	See the CIP Common Specification, Vol. 1, Chapter 4-9.1 for a description of these class attributes.						
8	Required	Set <sup>1</sup>	NV	Latch Input Error Time <sup>2</sup>	UINT	Any Safety Input or Test Output error will be latched for this minimum time. If the error is no longer present after this time the error condition may be reset by the module.	0 to 65565, in milliseconds. The default value is 1000

1. Applications including this attribute may restrict its access to Get
2. It is acceptable to support a subset of the complete range of values.

### 5-7.3 Instance Attributes

Table 5-7.2 Safety Discrete Input Point Instance Attributes

Attr ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
1	Optional	Get		Number of Attributes	USINT	Number Supported in this product.	
2	Optional	Get		Attribute List	Array of USINT	List of Attributes supported in this product.	
3	Optional	Get		Input Port Value	BOOL	Input point value at the terminal prior to the on/off delay evaluation.	0= Off 1= On  This is not safety data, there is no safety state.



**Safety Discrete Input Point Object, Class Code: 3D<sub>Hex</sub>**

Attr ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
4	Required	Get		Safety Status	BOOL	Input point status.	0= Alarm 1= OK This Value is Safety Data with a Safety State of 0.
5	Optional	Set <sup>1</sup>	NV	Off_On Delay <sup>2</sup>	UINT	Filter time for off to on transition 0-65,535 milli-seconds <sup>3</sup>	The default is 0.
6	Optional	Set <sup>1</sup>	NV	On_Off Delay <sup>2</sup>	UINT	Filter time for on to off transition 0-65,535 milli-seconds <sup>3</sup>	The default is 0.
7	Required	Get		Safety Input Logical Value	BOOL	Input point value after safety and on/off delay evaluation.	0= Off 1= On This Value is Safety Data with a Safety State of 0.
8	Optional	Set <sup>1</sup>	NV	Input Channel Mode <sup>2</sup>	USINT	Input Point Mode.	The default is 0. 0 = not used 1 = input, with test pulse from test out 2 = input, no dependency to test out (Used as, semiconductor input). 3 = input, no dependency to test out (Used as standard input).
9	Optional	Set <sup>1</sup>	NV	Test Source	USINT	A pointer to the test output that is used with this input when the Input Channel Mode is set to 1.	The instance number of the Discrete Output Point Object instance that is used as a test output for this input.
10	Optional	Get		Standard Input Value	BOOL	Input point value at the terminal with the on/off delay evaluation, but prior to safety input evaluation.	0= Off 1= On <i>This is not safety data, there is no safety state.</i>

1. Applications including this attribute may restrict its access to Get.
2. When this attribute is supported it is acceptable to support a subset of the complete range of values.
3. The input must be on for the filter time specified by the ON\_OFF Delay attribute before the OFF state is recorded in the Logical Value Attribute.

If the “Test Source” attribute of one or more Discrete Safety Input Point object instances is equal to an instance of the Discrete Output Point, and the “Input Channel Mode” attribute is equal to 1, then the “Test Output Mode” attribute (refer to Section 5-12-1, Attribute 13) of that Discrete Output Point instance must be set equal to 2= Test Output.

If the “Test Source” attribute of more than one Discrete Safety Input Point object instances is equal to an instance number of Discrete Output Point object, the “Input Channel Mode” attributes of those Safety Input Point instances must be identical.



#### **5-7.4 Common Services**

The Safety Discrete Input Point Object provides the following services.

**Table 5-7.3 Safety Discrete Input Point Common Services**

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
0E <sub>hex</sub>	Conditional	Required	Get_Attribute Single	Returns the contents of the specified attribute.
10 <sub>hex</sub>	Optional	Optional	Set_Attribute Single	Modifies an attribute value.

#### **5-7.5 Object-specific Services**

The Safety Discrete Input Point Object requires no Object-specific services.



## 5-7.6 Behavior

### 5-7.6.1 Type of Safety Inputs

The following table defines the types of safety inputs that are supported by the Safety Discrete Input Point Object. The effects on the Safety Input Logical Value, Status, and Test Source attributes are also shown. The SDIP object is designed for De-energized to trip behavior. Because of this design the Safety State of the Safety Data attributes is 0 or de-energized.

Table 5-7.4 Supported Safety Input Types

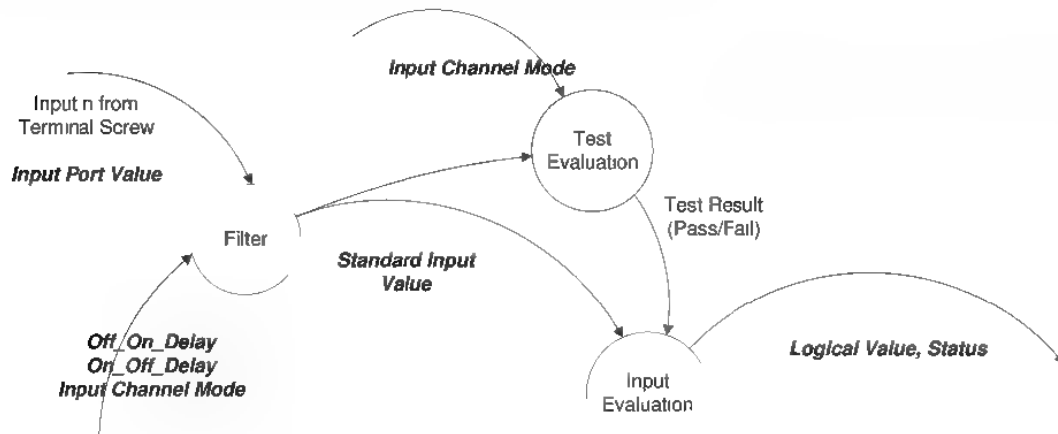
Input Channel Mode		Safety Input Type
<b>0 = not used</b> <b>1 = input, with test pulse from test out</b> <b>2 = input, no dependency to test out.</b> <b>(Used as, semiconductor input).</b> <b>3 = input, no dependency to test out.</b> <b>(Used as standard input).</b>		
0	Type	Not Used
	Input Port Value	Set to 0
	Standard Input Value	Set to 0
	Safety Input Logical Value	Set to 0
	Status	Set to 0
1	Type	<b>Single Channel Input (w/test pulses from test out)</b>
	Input Port Value	Set to Input Terminal value
	Standard Input Value	Set to Input Terminal value with the on/off delay evaluation
	Safety Input Logical Value	Set to Standard Input Value if no errors. Set to 0 if external or internal test fails
	Status	Set to Input Status
2	Type	<b>Single Channel Input (no dependency to test out.</b> <b>{Used in application as a semiconductor input})</b>
	Input Port Value	Set to Input Terminal value
	Standard Input Value	Set to Input Terminal value with the on/off delay evaluation
	Safety Input Logical Value	Set to Standard Input Value if no errors. Set to 0 if internal test fails.
	Status	Set to Input Status
3	Type	<b>Single Channel Input (no dependency to test out</b> <b>{Used in application as a standard input})</b>
	Input Port Value	Set to Input Terminal value
	Standard Input Value	Set to Input Terminal value with the on/off delay evaluation
	Safety Input Logical Value	Set to Standard Input Value if no errors. Set to 0 if optional internal test fails.
	Status	Set to Input Status



### 5-7.6.2 Safety Discrete Input Evaluation Behavior

The following diagram depicts the flow of the Safety Discrete Input Evaluation behavior.

**Figure 5-7.1 Safety Discrete Input Evaluation Flow**

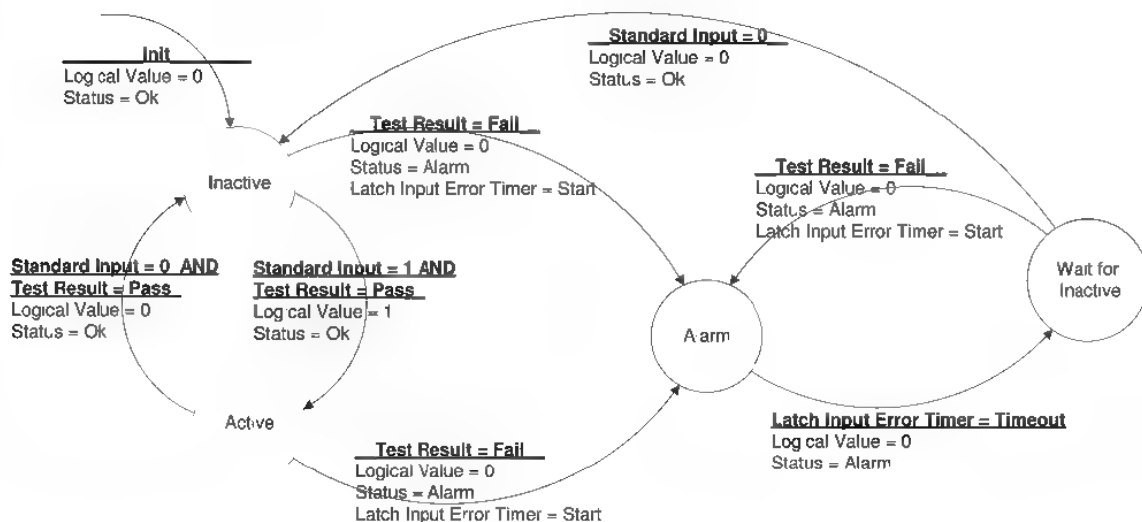


The details of the on/off delay filter algorithm used are product specific. The *Input Port Value* attribute shall be set to 0 when the *Input Channel Mode* attribute is set to Not Used.

The details of the Test Evaluation algorithms used are product specific.

The following diagram defines the behavior of the Input Evaluation portion of the diagram above.

### Figure 5-7.2 Input Evaluation Behavior





## 5-8 Safety Discrete Input Group (SDIG)

### Class Code: 3E<sub>hex</sub>

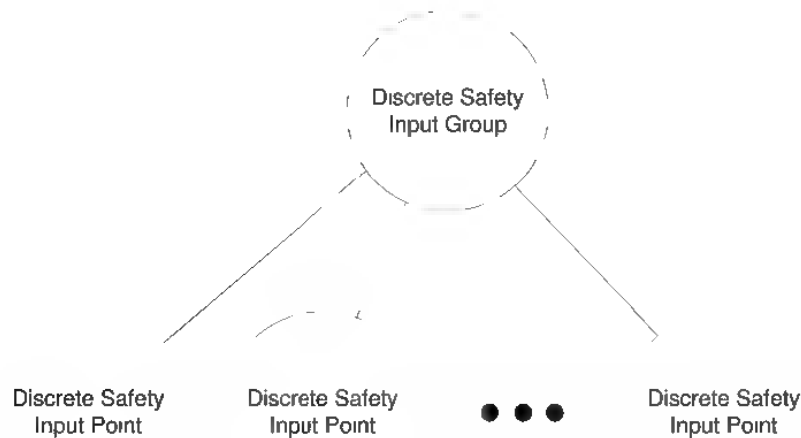
The Safety Discrete Input Group Object binds a group of discrete safety input points in a module. All points bound to the group share all attributes contained in the group. If an attribute is shared (points have the same attributes and the same attribute values) across more than one Safety Discrete Input Point, then that attribute can be contained in a Safety Discrete Input Group. A Discrete Input Point can be bound to more than one Discrete Input Group.

You must establish the list of discrete safety input points bound to the group at power-up, as the binding list is static and is a Get-only attribute of the group.

Use the Safety Discrete Input Group Object:

- when a single attribute is shared among many safety input points.
- to more efficiently access data (services that affect all members of a group can be supported by the Safety Discrete Input Group).

**Figure 5-8.1 Discrete Input Group**



### 5-8.1 Revision History

Revision	
01	Initial definition at First Release of the Object

### 5-8.2 Class Attributes

**Table 5-8.1 Safety Discrete Input Group Class Attributes**

Attr ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
1 thru 7	See the CIP Common Specification, Vol. 1, Chapter 4-9.1 for a description of the optional, reserved class attributes.						



### 5-8.3 Instance Attributes (Common)

**Table 5-8.2 Safety Discrete Input Group Instance Attributes**

Attr ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
1	Optional	Get		Number of Attributes	USINT	Number Supported in this product.	
2	Optional	Get		Attribute List	Array of USINT	List of Attributes supported in this product.	
3	Optional	Get		Number of Bound Instances	USINT	Number of points bound to this group	
4	Optional	Get		Binding	Array of UINT	List of all points bound to this group Instance ID	Class SDIP Instance #X Class SDIP Instance #Y...
5	Required	Get		Safety Status	BOOL	Status for all discrete input points in the group	0= Alarm 1= OK <i>This Value is Safety Data with a Safety State of 0.</i>
6	Optional	Set	NV	Off_On Delay <sup>1</sup>	UINT	Filter time for off to on transition 0-65,535 milli-seconds	The default is 0.
7	Optional	Set	NV	On_Off Delay <sup>1</sup>	UINT	Filter time for on to off transition 0-65,535 milli-seconds	The default is 0.

1. When this attribute is supported it is acceptable to support a subset of the complete range of values.

### 5-8.4 Common Services

The Safety Discrete Input Group Object provides the following services.

**Table 5-8.3 Safety Discrete Input Group Common Services**

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
0E <sub>hex</sub>	Conditional	Required	Get_Attribute_Single	Returns the contents of the specified attribute.
10 <sub>hex</sub>	Optional	Optional	Set_Attribute_Single	Modifies an attribute value.

### 5-8.5 Object-specific Services

The Safety Discrete Input Group Object requires no Object-specific services.

### 5-8.6 Behavior

The Status behavior is a simple AND'ing of the Status of the bound Safety Discrete Input Points.



## 5-9 Safety Discrete Output Point (SDOP)

### Class Code: 3B<sub>hex</sub>

The Safety Discrete Output Point Object models discrete safety outputs in a product. This object can be used in any device that contains discrete safety outputs. Note that the term “output” is defined from the network’s point of view. An output will consume data from the network.

The Safety Discrete Input Point interface is to real safety output points such as a Safety Semiconductor Outputs or a Safety Relays. The output is read from the object’s Safety Output Value attribute and applied to the safety output.

### 5-9.1 Revision History

Revision	
01	Initial definition at First Release of the Object

### 5-9.2 Class Attributes

**Table 5-9.1 Safety Discrete Output Point Class Attributes**

Attr. ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
1 thru 7	See the CIP Common Specification, Vol. 1, Chapter 4-9.1 for a description of the optional, reserved class attributes.						
8	Required	Set <sup>1</sup>	NV	Latch Output Error Time <sup>2</sup>	UINT	Any Safety Output error will be latched for this minimum time. If the error is no longer present after this time the error condition may be reset by the module.	0 to 65565, in milli-seconds. The default value is 1000.

1. Applications including this attribute may restrict its access to Get.

2. When this attribute is supported it is acceptable to support a subset of the complete range of values.

### 5-9.3 Instance Attributes (Common)

**Table 5-9.2 Safety Discrete Output Point Instance Attributes**

Att. ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
1	Optional	Get		Number of Attributes	USINT	Number Supported in this product.	
2	Optional	Get		Attribute List	Array of USINT	List of Attributes supported in this product.	
3	Required	Get <sup>1</sup>		Safety Output Value	BOOL	Output point value. This is the value that the output is commanded to be in.	0= Off 1= On <i>This Value is Safety Data with a Safety State of 0.</i>



**Volume 5: CIP Safety, Chapter 5: Object Library**  
**Safety Discrete Output Point Object, Class Code: 3B<sub>Hex</sub>**

Att. ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
4	Required	Get		Output Monitor Value	BOOL	Output point value, For semi-conductor outputs, the value at the terminal screw. For safety relay outputs it is the relay state red back from the internal auxiliary contacts.	0= Off 1= On <i>This is not safety data, there is no safety state.</i>
5	Required	Get		Safety Status	BOOL	Safety output point status	0= alarm 1= OK <i>This Value is Safety Data with a Safety State of 0.</i>
6	Optional	Set <sup>2</sup>	NV	Output Channel Mode <sup>3</sup>	USINT		0– Not Used 1= Used, No Test Pulses 2= Used, Test Pulses Default = Not Used

1. Settable only by Safety I/O Connections, not settable by explicit messaging
2. Applications including this attribute may restrict its access to Get
3. When this attribute is supported it is acceptable to support a subset of the complete range of values.

#### 5-9.4 Common Services

The Safety Discrete Output Point Object provides the following services.

**Table 5-9.3 Safety Discrete Output Point Common Services**

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
0E <sub>hex</sub>	Conditional	Required	Get_Attribute_Single	Returns the contents of the specified attribute.
10 <sub>hex</sub>	Optional	Optional	Set_Attribute_Single	Modifies an attribute value.

#### 5-9.5 Object-specific Services

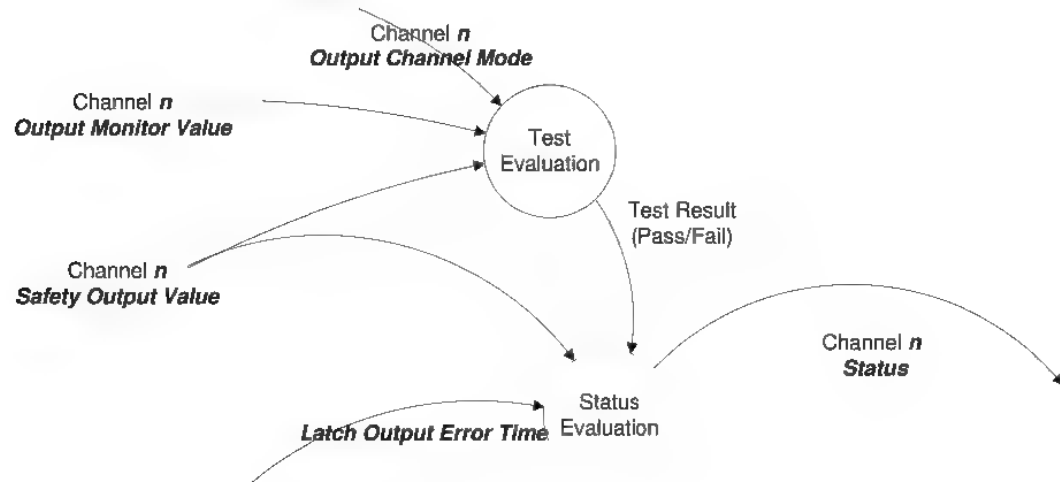
The Safety Discrete Output Point Object requires no Object-specific services.

#### 5-9.6 Behavior

Figure 5-9.1 depicts the flow of the Safety Discrete Output Evaluation behavior. The SDOP object is designed for De-energized to trip behavior. Because of this the Safety State of the Safety Data attributes is 0 or de-energized.



**Figure 5-9.1 Safety Discrete Output Evaluation Data Flow**

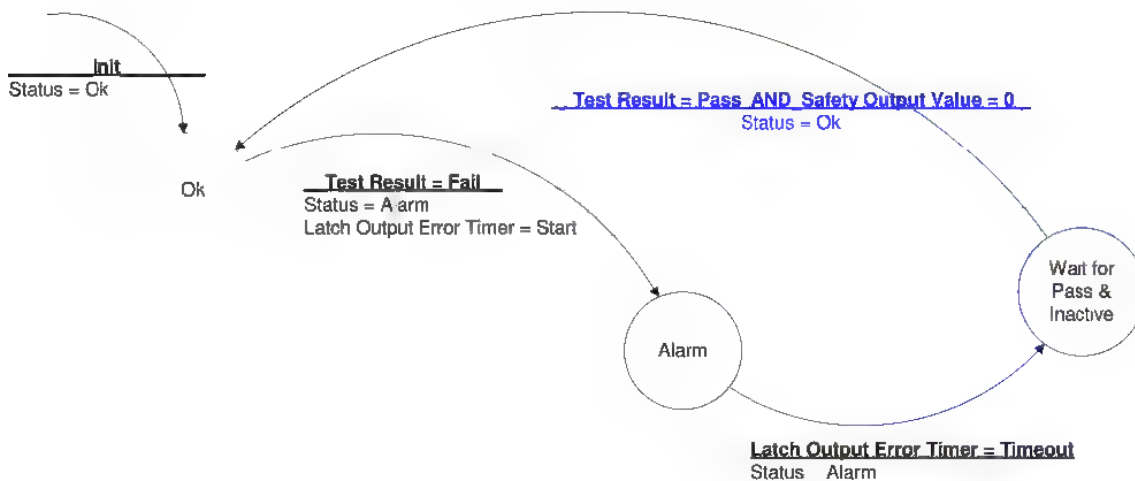


The details of the Test Evaluation algorithms used are product specific.

The Output should be de-energized any time that the Status indicates Alarm.

Figure 5-9.2 defines the behavior of the Status Evaluation portion of the Discrete Safety Output Evaluation behavior.

**Figure 5-9.2 Safety Discrete Output Status Evaluation Data Flow**





## 5-10 Safety Discrete Output Group (SDOG)

### Class Code: 3C<sub>Hex</sub>

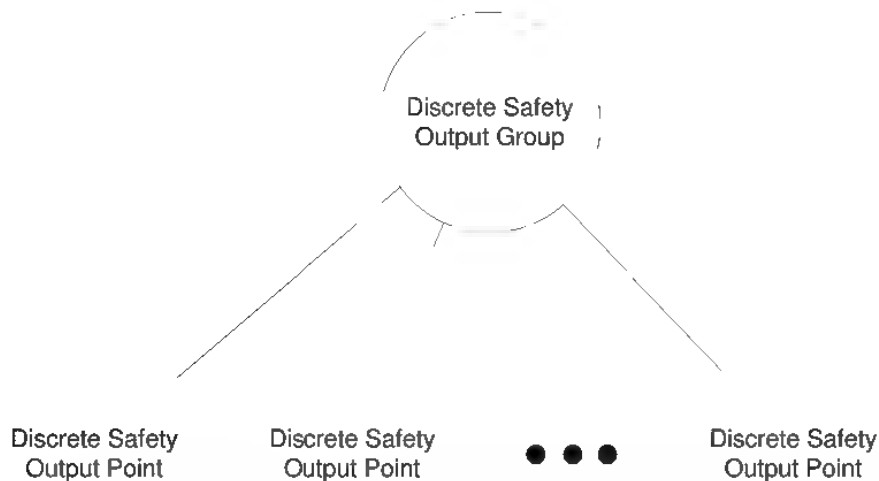
The Safety Discrete Output Group Object binds a group of discrete safety output points in a module. All points bound to the group share all attributes contained in the group. If an attribute is shared (points have the same attributes and the same attribute values) across more than one Safety Discrete Output Point, then that attribute can be contained in a Safety Discrete Output Group. A Safety Discrete Output Point can be bound to more than one Safety Discrete Output Group.

You must establish the list of discrete safety output points bound to the group at power-up, as the binding list is static and is a Get-only attribute of the group.

Use the Safety Discrete Output Group Object:

- when a single attribute is shared among many safety output points.
- to more efficiently access data (services that affect all members of a group can be supported by the Safety Discrete Output Group).

**Figure 5-10.1 Safety Discrete Output Group**



### 5-10.1 Revision History

Revision	
01	Initial definition at First Release of the Object



## 5-10.2 Class Attributes

Table 5-10.1 Safety Discrete Output Group Class Attributes

Attr ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
1 thru 7	See the CIP Common Specification, Vol. 1, Chapter 4-9 1 for a description of the optional, reserved class attributes						

## 5-10.3 Instance Attributes (Common)

Table 5-10.2 Safety Discrete Output Group Instance Attributes

Attr ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
1	Optional	Get		Number of Attributes	USINT	Number Supported in this product.	
2	Optional	Get		Attribute List	Array of USINT	List of Attributes supported in this product.	
3	Optional	Get		Number of Bound Instances	USINT	Number of points bound to this group	
4	Optional	Get		Binding	Array of UINT	List of all points bound to this group Instance ID	Class SDOP Instance #X Class SDOP Instance #Y...
5	Required	Get		Safety Status	BOOL	Safety output status for all discrete safety outputs in the group.	0= alarm 1= OK <i>This Value is Safety Data with a Safety State of 0</i>
7	Optional	Set	NV	Output Channel Mode <sup>1</sup>	USINT	Output Channel Mode for all discrete safety outputs in the group.	0= Not Used 1= Used, No Test Pulses 2= Used, Test Pulses

1. When this attribute is supported it is acceptable to support a subset of the complete range of values.

## 5-10.4 Common Services

The Safety Discrete Input Point Object provides the following services.

Table 5-10.3 Safety Discrete Output Group Common Services

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
0E <sub>hex</sub>	Conditional	Required	Get_Attribute_Single	Returns the contents of the specified attribute.
10 <sub>hex</sub>	Optional	Optional	Set_Attribute_Single	Modifies an attribute value.



**5-10.5 Object-specific Services**

The Safety Discrete Output Group Object requires no Object-specific services.

**5-10.6 Behavior**

The Status behavior is a simple AND'ing of the Status of the bound Safety Discrete Output Points.



## 5-11 Safety Dual Channel Output Object (SDCO)

### Class Code: 3F<sub>hex</sub>

The Safety Dual Channel Output Object models a pair of safety output channels as dual channel safety outputs in a product. This object must be used in conjunction with two instances of a Safety Discrete Output Point Object. This object can be used in any device that contains discrete safety outputs. Note that the term “output” is defined from the network’s point of view. An output will consume data on the network.

### 5-11.1 Revision History

Revision	
01	Initial definition at First Release of the Object

### 5-11.2 Class Attributes

**Table 5-11.1 Safety Dual Channel Output Class Attributes**

Attr ID	Need in Implem	Access Rule	N V	Name	Data Type	Description of Attribute	Semantics of Value
1 thru 7	See the CIP Common Specification, Vol 1, Chapter 4-9 1 for a description of the optional, reserved class attributes						

### 5-11.3 Instance Attributes (Common)

**Table 5-11.2 Safety Dual Channel Output Instance Attributes**

Attr ID	Need in Implem	Access Rule	N V	Name	Data Type	Description of Attribute	Semantics of Value
1	Optional	Get		Number of Attributes	USINT	Number Supported in this product.	
2	Optional	Get		Attribute List	Array of USINT	List of Attributes supported in this product.	
3	Required	Set <sup>1</sup>	N V	Dual Channel Mode	USINT	Mode of the dual channel safety output pair.	Default = 1. 0 – single channel 1 = dual channel
4	Optional	Set <sup>2</sup>	N V	Member One	USINT	A pointer to the first member of the dual channel safety output pair.	The instance number of the SDOP Object that is used as member one of the dual channel safety output pair.
5	Optional	Set <sup>2</sup>	N V	Member Two	USINT	A pointer to the second member of the dual channel safety output pair.	The instance number of the SDOP Object that is used as member one of the dual channel safety output pair.

- Applications including this attribute may restrict its access to Get.
- It is optional that these parameters be made publicly visible. However, all Validator instances shall have internal parameters which represent these variables.



#### 5-11.4 Common Services

The Safety Dual Channel Output Object provides the following services.

**Table 5-11.3 Safety Dual Channel Output Common Services**

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
0E <sub>hex</sub>	Conditional	Required	Get_Attribute_Single	Returns the contents of the specified attribute.
10 <sub>hex</sub>	Optional	Optional	Set_Attribute_Single	Modifies an attribute value.

#### 5-11.5 Object-specific Services

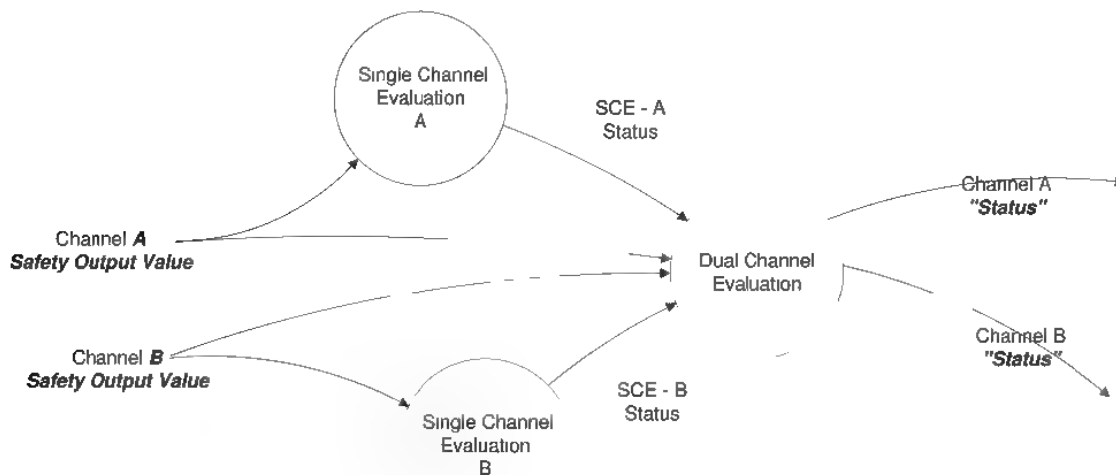
The Safety Dual Channel Output Point Object requires no Object-specific services.

#### 5-11.6 Dual Channel Evaluation Behavior

The Safety Dual Channel Output has no behavior above the Safety Discrete Output Point (SDIP) object instances when the Output Channel Mode is set to Single.

Figure 5-11.1 depicts the flow of the Dual Channel Evaluation behavior for Output Channel Mode set to Dual.

**Figure 5-11.1 Dual Channel Evaluation Data Flow**



The behavior is described in terms of:

1. Single Channel Evaluation of Channels A and B
2. Dual Channel Evaluation

The Single Channel Evaluation is described by the Safety Discrete Output Point (SDOP) object for all outputs used in the Safety Dual Channel Output (SDCO).



**Volume 5: CIP Safety, Chapter 5: Object Library**  
**Safety Dual Channel Output Group Object, Class Code: 3F<sub>Hex</sub>**

From a behavioral perspective, the Dual Channel Evaluation portion of the SDCO takes the following inputs:

- A. The Status from Channel A Single Channel Evaluation (Ok or Alarm)
- B. The Status from Channel B Single Channel Evaluation (Ok or Alarm)
- C. The Safety Out Values for Channel A (0 or 1)
- D. The Safety Out Values for Channel B (0 or 1)

From a behavioral perspective, the Dual Channel Evaluation produces the following outputs:

- The *Status* attributes of the Channel A and Channel B Safety Discrete Output Point instances.

Table 5-11.4 defines the Dual Channel Evaluation behavior for Output Channel Mode set to Dual.

**Table 5-11.4 Dual Channel Evaluation Behavior**

From Single Channel Evaluation		Safety Output Values		Results applied to Channel A and Channel B “Status” attributes
Status A	Status B	Channel A	Channel B	“Status” A & B
Ok	Ok	0	0	OK
Ok	Ok	0	1	Alarm
Ok	Ok	1	0	Alarm
Ok	Ok	1	1	OK
Alarm	x	x	x	Alarm
x	Alarm	x	x	Alarm

x = don't care

The dual channel output status bits may only transition from Alarm to Ok when both outputs are in the inactive state at the same time.



## 5-12 Discrete Output Point Object

### Class Code: 09<sub>hex</sub>

This section defines the extensions to the Discrete Output Point (DOP) Object, Class 0x09 for CIP Safety. (Also see Volume 1, Chapter 5 for the complete object definition.)

### 5-12.1 Test Output Mode for Safety Attribute

The Test Output Mode For Safety attribute is added to the DOP object.

**Table 5-12.1 Test Output Attribute Detail**

Att. ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
13	Optional1	Set	NV	Test Output Mode When this attribute is supported it is acceptable to support a subset of the complete range of values.	USINT	Mode of this Test output Point	The default is 0. See Semantics

1. When this attribute is supported it is acceptable to support a subset of the complete range of values.

### 5-12.2 Test Output Mode Semantic

The Test Output Mode attribute may be used on modules that have output points that are configurable as either standard Discrete Output Points or as Test Output Points to be used within a safety system. The following table defines the semantic of the test output mode.

**Table 5-12.2 Test Output Mode Values**

Test Output Point Mode	Test Output Point Type
0	Not Used, The output is deactivated. Default Value
1	The output point is configured as a standard output. The Output is controlled by the Value attribute.
2	The output point is configured as a test output. This test output is used in conjunction with the Safety Discrete Input Point object instance(s) whose Test Source attribute (refer to Section 5-7.3, Attribute 9) is equal to this Output Points instance number. The Output is activated with test pulses and cannot be controlled externally.
3	The output point is configured as a power supply output. The output is permanently activated or set to high
4	The output point is configured as a Muting Lamp output. The status of the Muting Lamp Output is available in the Status attribute of this DOP instance. The Output is controlled by the Value attribute.
All other values	Reserved by CIP



## 5-13 TCP/IP Object

### Class Code: F5 Hex

This section defines the changes and additions required of the TCP/IP object to support CIP safety.

### 5-13.1 TCP/IP Object Instance Attribute Changes

An instance attribute shall be added to allow the Safety Network Number (SNN) to be read. The definition for this attribute is as follows:

**Table 5-13.1 Safety Network Number Attribute Detail**

Attr ID	Need in Implem	Access Rule	Name	Data Type	Description of Attribute	Semantics of Value
7	Required	Get	Safety Network Number	6 octets	System-wide unique number for the subnet this device resides on.	Attribute shall contain the SNN portion of the TUNID in the Safety Supervisor. See Chapter 2-3.1.1 for a complete definition of the SNN.

SRS209 Devices implementing EtherNet/IP Safety shall support attribute 7 of the TCP/IP object.



## 5-14 Safety Analog Input Point Object

### Class Code: 49<sub>hex</sub>

The Safety Analog Input Point (SAIP) Object models Safety Analog Input channels in a product. It can be used in applications as simple as a single analog point and as complex as an analog I/O control module. Note that the term “input” is defined from the network’s point of view. An input produces data onto the network.

The Safety Analog Input Point interfaces to analog input devices such as a temperature sensor or pressure transducer. The input is sampled and processed through filtering and conditioning algorithms and the resulting data is produced onto the network. When signal integrity is verified by vendor specific algorithms, the result is saved in this object’s Safety Analog Input Value attribute. The main feature of the Safety Analog Input Point is that underlying device implements internal diagnostics sufficient to ensure that the Safety Analog Input Value meets Safety Integrity Level (SIL) requirements (SIL level achieved depends on the implementation). Whenever a fault is detected that makes the signal integrity unreliable, a fault state is declared and a safe value is reported to the control system.

### 5-14.1 Revision History

This is the initial release of this object. The table below represents the revision history:

**Table 5-14.1 Safety Analog Input Point Object Revision History**

Revision	Reason for object definition update
01	Initial Definition at First Release of Specification

### 5-14.2 Class Attributes

**Table 5-14.2 Safety Analog Input Point Object Class Attributes**

Attr ID	Need in Implem	Access Rule	N V	Name	Data Type	Description of Attribute	Semantics of Value
1 thru 7	See the CIP Common Specification, Vol. 1, and Chapter 4-9.1 for a description of the optional, reserved class attributes.						

### 5-14.3 Instance Attributes

**Table 5-14.3 Safety Analog Input Point Object Instance Attributes**

Attr ID	Need in Implem	Access Rule	NV	Attribute Name	Data Type	Description of Attribute	Semantics of Values
1	Optional	Set <sup>1</sup>	NV	Data Type	USINT	Determines the data type of Safety Analog Input Value and all related attributes as specified in this table.	C3 = INT (default) See Appendix C-6.1 See Semantics section.



Safety Analog Input Point Object, Class Code: 49<sub>Hex</sub>

Attr ID	Need in Implem	Access Rule	NV	Attribute Name	Data Type	Description of Attribute	Semantics of Values
2	Required	Get	V	Safety Analog Input Value	INT or based on <i>Data Type</i>	Analog Input value after conditioning, filtering and SIL integrity testing.	See Semantics section. This is Safety Data with a Safe Value defined by Safe State attributes.
3	Required	Set <sup>1</sup>	NV	Input Range	USINT	Input range the point is operating in	See Semantics section.
4	Required	Set	NV	Input Channel Mode	USINT	Input Point Mode	0 = Not used (default) 1 = Safety, with SIL integrity testing 2 = Standard, with no SIL integrity testing
5	Required	Get	V	Input Point Status	BOOL	Status of the Analog Input Point	0 = Alarm 1 = OK This is Safe Data with Safety State of 0
6	Required	Get	V	Point Fault Reason	USINT	Determines the fault type that exists on the input	1 = Operating Normally See Semantics section.
7	Optional	Set <sup>1</sup>	NV	Real Time Sample Period	UINT	Millisecond interval at which input data is sampled	See Semantics section.
8	Optional	Set <sup>1</sup>	NV	Input Error Latch Time	UINT	Minimum time a Safety Input error will be latched for.	0 – 65535 ms, default = 1000 See Semantics section.
9	Optional	Set	NV	Full Scale	INT or based on <i>Data Type</i>	The value of Full Scale for the sensor	The value of <i>Safety Analog Input Value</i> corresponding to the Full Scale calibrated measurement of the sensor. Default = Maximum value for the <i>Data Type</i> .
10	Optional	Set <sup>1</sup>	NV	Safe State Behavior	USINT	Defines behavior for value reporting when faulted.	0 = Use Safe State Value, default. See Semantics section.
11	Optional	Set <sup>1</sup>	NV	Safe State Value	INT or based on <i>Data Type</i>	Safe State Analog Input value.	0 = default.
12	Optional	Set	NV	Low Signal	INT or based on <i>Data Type</i>	Determines the data low signal value for scaling the raw analog input.	Default = Min value for <i>Data Type</i> . See Semantics section.
13	Optional	Set	NV	High Signal	INT or based on <i>Data Type</i>	Determines the data high signal value for scaling the raw analog input.	Default = Max value for <i>Data Type</i> . See Semantics section.
14	Optional	Set	NV	Low Engineering Value	INT or based on <i>Data Type</i>	Determines the data low engineering value for scaling the raw analog input.	Default = Input Range Min. See Semantics section



Safety Analog Input Point Object, Class Code: 49<sub>Hex</sub>

Attr ID	Need in Implem	Access Rule	NV	Attribute Name	Data Type	Description of Attribute	Semantics of Values
15	Optional	Set	NV	High Engineering Value	INT or based on <i>Data Type</i>	Determines the data high engineering value for scaling the raw analog input.	Default = Input Range Max. See Semantics section.
16	Optional	Get	V	Alarm and Warning Status	BYTE	Indicates the level of alarm or warning that has occurred.	0x0F = No alarms, See Semantics section.
17	Optional	Set	NV	Alarm Enable	BOOL	Enable Alarm Status Checking	0 = Disable, default 1 = Enable
18	Optional	Set	NV	Alarm Trip Point High	INT or based on <i>Data Type</i>	Specifies the <i>Value</i> above which an Alarm Condition will occur.	See Semantics section, Default = Max value for <i>Data Type</i>
19	Optional	Set	NV	Alarm Trip Point Low	INT or based on <i>Data Type</i>	Specifies the <i>Value</i> below which an Alarm Condition will occur	See Semantics section, Default = Min value for <i>Data Type</i>
20	Optional	Set	NV	Alarm Hysteresis	INT or based on <i>Data Type</i>	Specifies the amount by which the Safety Analog Input Value must recover to clear the Alarm condition.	0 = default See Semantics section.
21	Optional	Set	NV	Alarm Settling Time	UINT	Determines the time that the Value must exceed the Alarm Trip Point before an Alarm Condition will occur.	Time in milliseconds 0 = default See Semantics section
22	Optional	Set	NV	Warning Enable	BOOL	Enable Warning Status Checking	0 = Disable, default 1 = Enable
23	Optional	Set	NV	Warning Trip Point High	INT or based on <i>Data Type</i>	Specifies the <i>Value</i> above which a Warning Condition will occur.	See Semantics section, Default = Max value for <i>Data Type</i>
24	Optional	Set	NV	Warning Trip Point Low	INT or based on <i>Data Type</i>	Specifies the <i>Value</i> below which a Warning Condition will occur.	See Semantics section, Default = Min value for <i>Data Type</i>
25	Optional	Set	NV	Warning Hysteresis	INT or based on <i>Data Type</i>	Specifies the amount by which the Safety Analog Input Value must recover to clear the Warning condition	0 = default See Semantics section.
26	Optional	Set	NV	Warning Settling Time	UINT	Determines the time that the Value must exceed the Warning Trip Point before a Warning Condition will occur	Time in milliseconds 0 = default See Semantics section.



Safety Analog Input Point Object, Class Code: 49<sub>Hex</sub>

Attr ID	Need in Implem	Access Rule	NV	Attribute Name	Data Type	Description of Attribute	Semantics of Values
27	Optional	Get	NV	Over Range Value	INT or based on <i>Data Type</i>	Shows the highest valid Analog Input value after conditioning and filtering.	The value above which the Over Range condition is declared. See Semantics
28	Optional	Get	NV	Under Range Value	INT or based on <i>Data Type</i>	Shows the lowest valid Analog Input value after conditioning and filtering.	The value below which the Under Range condition is declared. See Semantics
29	Optional	Get	NV	Offset-A Data Type	USINT	Determines the data type of <i>Offset-A</i>	C3 = INT (default) See Appendix C-6.1 See Semantics section .
30	Optional	Set	NV	Offset-A	INT or based on <i>Offset-A Data Type</i>	Amount added before Gain to derive <i>Safety Analog Input Value</i>	0 = default See Semantics section.
31	Optional	Get	NV	Gain Data Type	USINT	Determines the data type of <i>Gain</i>	C3 = INT (default) See Appendix C-6.1 See Semantics section .
32	Optional	Set	NV	Gain	REAL or based on <i>Offset-A Data Type</i>	Amount scaled to derive <i>Safety Analog Input Value</i>	1.0 = default See Semantics section.
33	Optional	Get	NV	Unity Gain Reference	REAL or based on <i>Gain Data Type</i>	Specifies the value of the Gain attribute equivalent to a gain of 1.0	1.0 = default Used to normalize the <i>Gain</i> attribute.
34	Optional	Set	NV	Offset-B	INT or based on <i>Data Type</i>	Amount added to derive <i>Safety Analog Input Value</i>	0 = default See Semantics section
35	Optional	Set	NV	Produce Trigger Delta	INT or based on <i>Data Type</i>	Amount by which <i>Safety Analog Input Value</i> must change to cause a change of state production	0 = default See Semantics section.
36-79	Reserved						
80-96	Reserved for Subclass use						
97,98	Reserved						
99	Optional	Get	NV	Subclass	UINT	Identifies a subset of application specific attributes, services, and behaviors.	0 = No subclass (default) 1 – 65535 Reserved

1. Attribute may be implemented as Get only.



**Important:** Attributes are only settable when the Safety Supervisor is in the Configuring state.

#### **5-14.4 Semantics**

This section defines the semantics of the Safety Analog Input Point object attributes.

##### **5-14.4.1 Data Type, Attribute 1**

The Data Type attribute determines the data type to be used by the Safety Analog Input Value, High and Low Signal, High and Low Engineering Value, Full Scale, Safe State Value, Over Range, Under Range, and Alarm and Warning Trip Points, Hysteresis, Offset-B and Produce Trigger Delta attributes. The implementation may support only those Data Type values needed for its application. If Data Type is not supported, then the data type of the Safety Analog Input Value and all other related attributes shown above, defaults to INT.

The Data Type attribute uses the following enumerated numeric types defined in Volume 1, Appendix C-6.1.

- SINT (C2), USINT (C6)
- INT (C3), UINT (C7)
- DINT (C4), UDINT (C8)
- LINT (C5), ULINT (C9)
- REAL (CA) LREAL (CB)

##### **5-14.4.2 Safety Analog Input Value, Attribute 2**

The Safety Analog Input Value attribute represents the analog input signal after it has been processed through the conditioning, filtering and testing algorithms. This value is reported to the safety controller. In general, the SIL integrity testing covers the signal input circuit reliability and value reliability. The methods used to meet SIL integrity testing are vendor specific.

The Safety Analog Input Value is derived as follows:

$$\text{Value} = \text{GainN} * (\text{Raw Sensor Value} + \text{Offset-A}) + \text{Offset-B}$$

Where  $\text{GainN} = \text{Gain} / \text{Unity Gain Reference}$

The Offset-A and Offset-B values are typically modified by the Zero\_Adjust services and the Gain is modified by Gain\_Adjust service.

When the point input is paired with a second Safety Analog Input Point (see Safety Dual Channel Analog Input, CIP Safety Specification, Vol 5, section 5-TBD), the Safety Analog Input Value is the “consensus” value of the dual input pair. The details of developing the “consensus” value are vendor specific.



### **5-14.4.3 Input Range, Attribute 3**

The Input Range attribute determines the set of analog values within which the inputs may operate. The value of the Input Range affects the default and legal values that other attributes may have.

Input Range is required if the point may be switched to operate within different ranges, which changes the behavior of the point. For example, if the point is configured to operate from 0V to 5V or from 4mA to 20mA, the variability of ranges will likely change the expected behavior of the Value attribute and perhaps change vendor specific attributes. The implementation may support only those range values needed for its application.

**Table 5-14.4 Input Range Values**

Range Value	Range and Units
0	-10 to 10 volts
1	0 to 5 volts
2	0 to 10 volts
3	4 to 20 milliamps (default)
4	-15 to 75 milli volts
5	-15 to 30 milli volts
6	-5 to 5 volts
7	1 to 5 volts
8	0 to 20 milliamp
9	0 to 50 milliamp
10-99	Reserved
100-199	Vendor Specific
200-255	Reserved

### **5-14.4.4 Input Channel Mode, Attribute 4**

The Input Channel Mode attribute specifies how the analog input signal is used in the application.

- 0 – Input Signal Not Used. The input signal is not used and the Safe State Value is reported in the Safety Analog Input Value to the safety controller. No faults, alarms or warnings are evaluated. The Input Point Status data is set to zero. The Fault Reason value is set to 8 to indicate that the input point is not used.
- 1 – Safety with testing. The input signal is read from the terminal, processed through the conditioning and filtering algorithms and reported in the Safety Analog Input Value to the safety controller. The signal integrity is verified by the internal test algorithms. If any faults are detected, the Input Point Status is set to zero and the Safe State Value is reported in the Safety Analog Input Value to the safety controller. The Fault Reason value is set to a value that indicates what caused the fault.



- 2 – Standard. The input signal is read from the raw input and processed through the conditioning and filtering algorithms and reported in the Safety Analog Input Value to the safety controller. The safety verifications are applied to the input signal but the input point can not be paired with a safety input point in a dual channel configuration when “Standard” mode is selected.

#### **5-14.4.5 Input Point Status, Attribute 5**

The Input Point Status attribute provides a status of the safety analog input point. When set (=1), the Input Point is operating normally. When cleared (=0), a signal integrity related fault has been declared. The Point Fault Reason attribute provides detail about what event caused the fault.

#### **5-14.4.6 Point Fault Reason, Attribute 6**

The Fault Reason attribute provides a detailed description of what condition was detected that caused the Input Point Status attribute to be set to 0. If the Input Point Status is one, the Point Fault Reason is also one, Operating Normally.

**Table 5-14.5 Fault Reason Value Definitions**

Value	Definition
0	Reserved
1	Operating Normally
2	Over Range
3	Under Range
4	Signal Integrity Failure - The channel has failed a safety related integrity test
5	Discrepancy Error
6	Partner Discrepancy Error
7	Calibration Failure
8	Input Point Mode is “Input Signal not Used”
9-99	Reserved
100-199	vendor specific
200-255	Reserved

#### **5-14.4.7 5-11.5.7 Real Time Sample Period, Attribute 7**

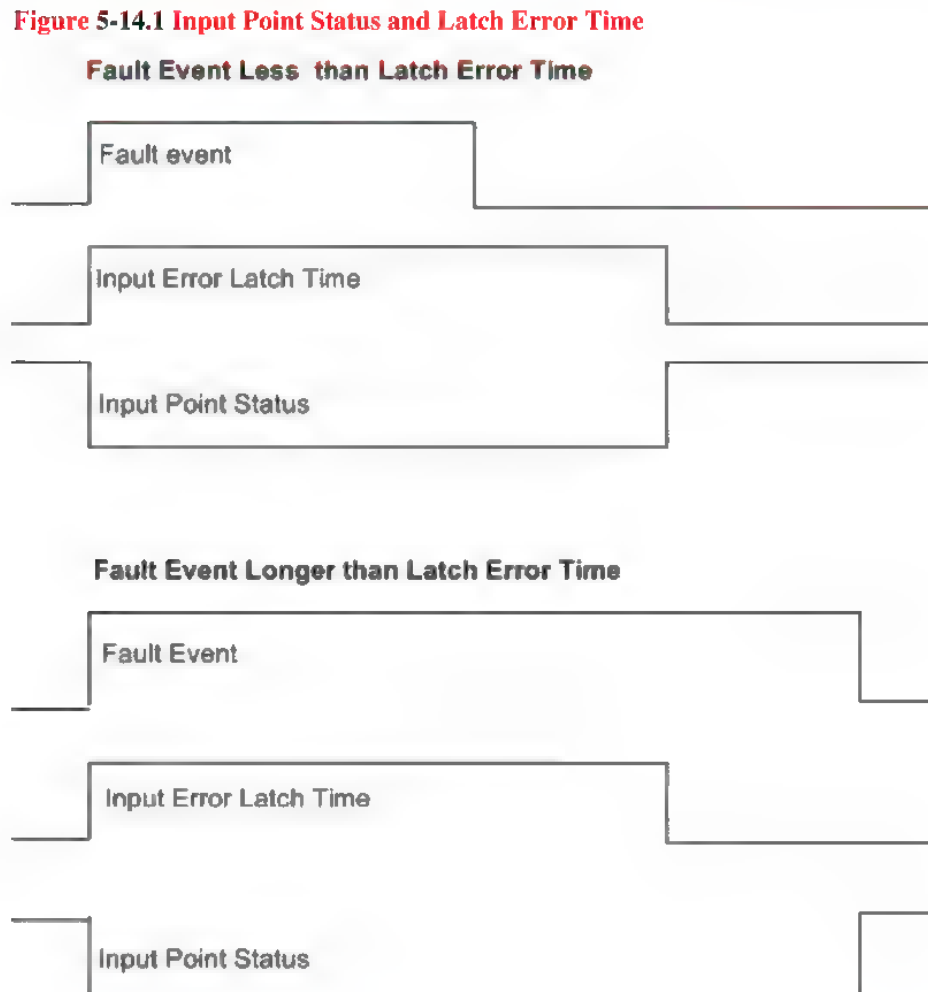
The Real Time Sample Period attribute specifies the time in milliseconds at which the application will sample the input data. This value is an internal time that is independent from any IO connection RPI. This value may be specified as a constant by the implementation. Vendors are free to limit the valid settings as required by the implementation.



#### **5-14.4.8    5-11.5.8    Input Error Latch Time, Attribute 8**

The Input Error Latch Time specifies the minimum time that an input point fault shall be latched after the fault condition is detected. If the fault condition clears before the latch time expires, the Input Point Status value shall be held at zero and the Point Fault Reason is held at the associated value for the latch time duration. If the error condition is cleared after the latch timer expires, the Input Point Status and Point Fault Reason are set to one by the module. A value of zero specifies that no latching will occur.

The following figure illustrates the relationship of the fault event and the Input Error Latch Time and the Input Point Status value.



#### **5-14.4.9    Full Scale, Attribute 9**

Value returned to the controller when the Safe State Behavior attribute is set to “Full Scale” (1). The Full Scale Value is determined by the Data Type and Input Range. Default value is the maximum value of the data type.



#### **5-14.4.10 Safe State Behavior, Attribute 10**

The Safe State Behavior attribute specifies the value that the Safety Analog Input Point should report in the Safety Analog Input Value to the safety controller whenever the input channel is in the “Safe State”, e.g. a fault in the input or power circuitry is detected.

**Table 5-14.6 Safe State Behavior Definitions**

Value	Definition
0	Use Safe State Value (Default)
1	Full Scale
2	Hold Last Value
3	Continue Sensing
4-99	Reserved
100-255	Vendor Specific

#### **5-14.4.11 Safe State Value, Attribute 11**

The Safe State Value attribute represents the analog input value that is reported in the Safety Analog Input Value to the safety controller whenever the Input Point Status is 0 and the Safe State Behavior is Use Safe State Value. The default is zero.

#### **5-14.4.12 High and Low Signal, High and Low Engineering Values, Attributes 12-15**

The Safety Analog input Object derives a reading from a physical sensor. The reading is converted to the Data Type and Input Range (units) specified for the Analog Input Value attribute. The High Signal, Low Signal, High Engineering and Low Engineering attribute values are applied to the sensor reading as specified by the following formula.

$$\text{Value} = (\text{Raw Input Value} - \text{Low Signal}) * \text{Adjust} + \text{Low Engineering}$$

$$\text{Adjust} = (\text{High Engineering} - \text{Low Engineering}) / (\text{High Signal} - \text{Low Signal})$$

The High and Low Signal attributes may be constant, non-settable values depending on the implementation.

#### **5-14.4.13 Alarm/Warning Status, Attributes 16**

The Alarm/Warning Status attribute is a bit mapped byte that represents the alarm and warning status of the Safety Analog Input. The following bits are defined.

**Table 5-14.7 Alarm/Warning Status Bit Definition**

Bit	Definition
0	High Alarm Exception 0 = Fault, 1 = Within range
1	Low Alarm Exception 0 = Fault, 1 = Within range
2	High Warning Exception 0 = Fault, 1 = Within range
3	Low Warning Exception 0 = Fault, 1 = Within range
4-7	Reserved



#### **5-14.4.14 Alarm/Warning Enable, Attributes 17, 22**

The Alarm and Warning Enable attributes allow the user to specify when the process alarms and/or warnings are to be monitored. When set to 1, the alarms and warnings are monitored and the associated status bits in the Alarm and Warning Status attribute are set to one or zero, as appropriate depending on the alarm or warning trip point, hysteresis and settling time attributes.

#### **5-14.4.15 Alarm/Warning Trip Points, Hysteresis and Settling Time, Attributes 18-21, 23-26**

The Alarm or Warning Trip Point High attribute is the level above which an alarm or warning will be declared when the Safety Analog Input Value exceeds this level.

The Alarm or Warning Trip Point Low attribute is the level below which an alarm or warning will be declared when the Safety Analog Input Value is less than this level.

Before the Alarm and Warning configuration is applied, the values shall be validated to ensure that:

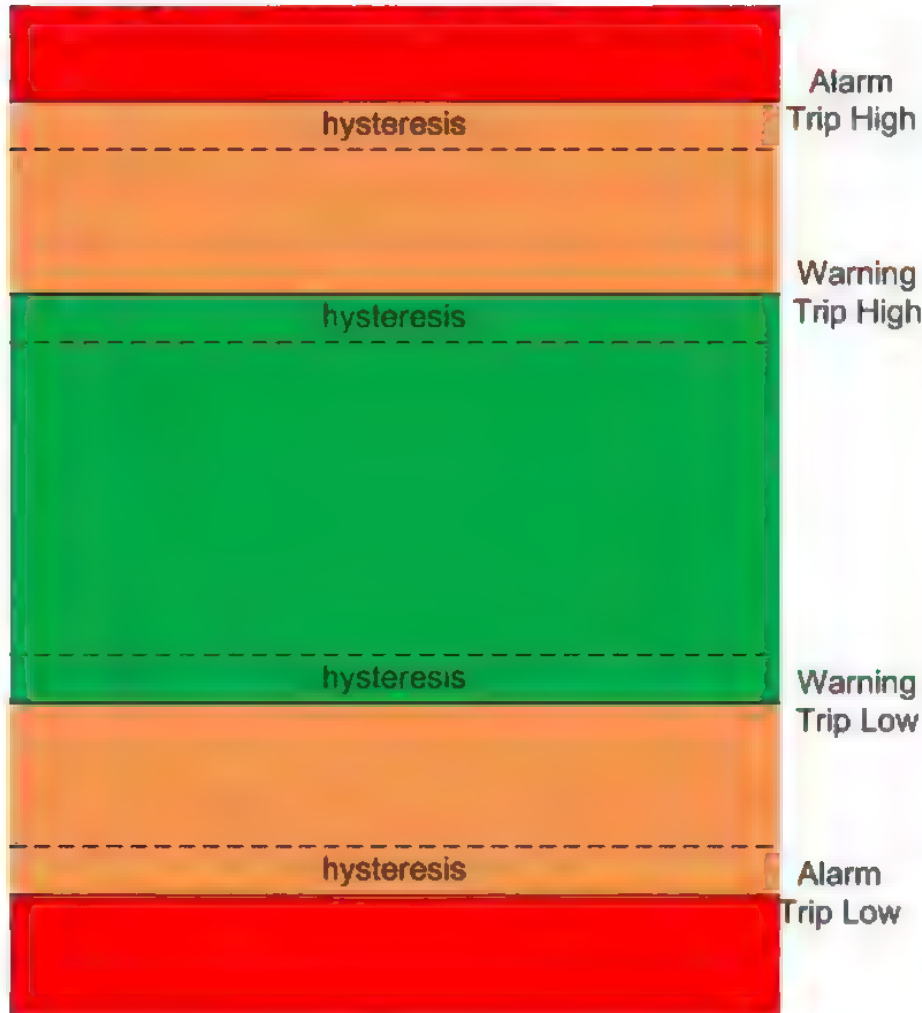
- Alarm Trip Point High  $\geq$  Warning Trip Point High
- Alarm Trip Point Low  $\leq$  Warning Trip Point Low
- Alarm Trip Point High  $>$  Alarm Trip Point Low
- Warning Trip Point High  $>$  Warning Trip Point Low

The Alarm or Warning Hysteresis attribute specifies the amount by which the Safety Analog Input Value must transition in order to clear an alarm or warning condition. For example, when the Alarm Trip Point High value is 1000 and the Alarm Hysteresis is 2, an alarm will be declared then the Safety Analog Input Value exceeds 1000 and will be cleared when the value drops below 998. Conversely, when the alarm trip point low value is 100, an alarm will be declared then the Safety Analog Input Value drops below 100 and will be cleared when the value recovers to 102.

The Alarm or Warning Settling Time attribute specifies the amount of time that the Safety Analog Input Value attribute must exceed the Alarm or Warning Trip Point level before an alarm or warning is declared. The Alarm or Warning Settling Time also applies to the clearing of the alarm or warning. When the input value returns to within the specified alarm or warning value for the Alarm or Warning Settling Time, the affected alarm or warning bit shall be set to zero.



Figure 5-14.2 Alarm and Warning Levels and Hysteresis



#### 5-14.4.16 Over Range Value, Attribute 27

The Over Range Value attribute represents the highest valid analog input signal after it has been processed through the conditioning and filtering algorithms. The value may change based on scaling factor. The default is product specific. If this value is exceeded, the Input Point Status is set to 0 and the Point Fault Reason is set to 1 (Over Range).

#### 5-14.4.17 Under Range Value, Attribute 28

The Under Range Value attribute represents the lowest valid analog input signal after it has been processed through the conditioning and filtering algorithms. The value may change based on scaling factor. The default is product specific. If this value is exceeded, the Input Point Status is set to 0 and the Point Fault Reason is set to 2 (Under Range).



#### **5-14.4.18 Offset-A and Gain Data Type, Attributes 29, 31**

The Offset A Data Type attribute determines the data type to be used by the Offset A attribute. The implementation may support only those Data Type values needed for its application. If Offset-A Data Type is not supported, then the Offset-A data type defaults to INT.

The Offset-A Data Type attribute uses the enumerated numeric types defined in section 5-14.4.1, Data Type, Attribute 1.

#### **5-14.4.19 Offset-A and B, Gain and Unity Gain Reference, Attributes 30, 32 - 34**

The Offset-A and Offset-B and Gain attributes are values used to calculate the Safety Analog Input Value as shown in section 5-14.4.2.

#### **5-14.4.20 Produce Trigger Delta, Attribute 35**

This attribute is used in conjunction with a “Change of State” connection. It specifies the amount of change in the Safety Analog Input Value that triggers a Change of State IO production. Each time a Change of State production is generated, the value of the Safety Analog Input Value attribute is saved and used with the Produce Trigger Delta value to determine the next data production. The Produce Trigger Delta is an absolute value.

#### **5-14.4.21 Subclass, Attribute 99**

Each instance level subclass defines an application specific (e.g. temperature sensor or flame detection sensor) set of attributes, services and behaviors. The range for subclass definitions starts at 96 and numbers downward for attributes and 0x63 and numbers downward for object specific services. The subclass is defined by the value of its subclass instance attribute. Subclass specifications will be added to this object definition as needed.

### **5-14.5 Common Services**

The Safety Analog Input Object provides the following Common Services:

**Table 5-14.8 Safety Analog Input Point Object Common Services**

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
0Ehex	Required	Required	Get_Attribute_Single	Returns the contents of the specified attribute.
10hex	n/a	Conditional <sup>1</sup>	Set_Attribute_Single	Modifies an attribute value.
01hex	Optional	Optional	Get_Attributes_All	Returns a predefined listing of this objects attributes (See the Get_Attributes_All Response definition below)
02hex	n/a	Optional	Set_Attributes_All	Modifies the contents of the attributes of the class or object.
1. The Set_Attribute_Single service is required if settable Instance Attributes are implemented.				
See Appendix A for definition of these services				



### 5-14.5.1 Get\_Attributes\_All Response

At the Class level, the order of the attributes returned in the “Object/service specific reply data” portion of the Get\_Attributes\_All response is defined in Volume 1, Chapter 4-5.1.

At the Instance level, the order of the attributes returned in the “Object/service specific reply data” portion (see Chapter 4 for a description of the Service Data field) of the Get\_Attributes\_All response is as follows:

**Table 5-14.9 Get\_Attributes\_All Response Data – Instance Level**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Value Data Type Default = C3							
1	Safety Analog Input Value (low byte)							
n+1	Safety Analog Input Value (high byte)							
.	Input Range Default = 3							
.	Input Channel Mode Default = 0							
.	0	0	0	0	0	0	0	Input Point Status Default = 1
.	Point Fault Reason Default = 1							
.	Real Time Sample Period (low byte) Default = 0							
.	Real Time Sample Period (high byte)							
.	Input Error Latch Time (low byte) Default = 0							
.	Input Error Latch Time (high byte)							
.	Full Scale (low byte) Default = Data Type Maximum							
.	Full Scale (high byte)							
.	Safe State Behavior Default = 0							
.	Safe State Value (low byte) Default = 0							
.	Safe State Value (high byte)							
.	Low Signal (low byte) Default = Data Type minimum							
.	Low Signal (high byte)							
.	High Signal (low byte) Default = Data Type maximum							
.	High Signal (high byte)							
.	Low Engineering (low byte) Default = Data Type minimum							
.	Low Engineering (high byte)							
.	High Engineering (low byte) Default = Data Type maximum							
.	High Engineering (high byte)							
.	Alarm and Warning Status Default = 0x0F							
.	Alarm Enable Default = 0							
.	Alarm Trip Point High (low byte) Default = Data Type maximum							
.	Alarm Trip Point High (high byte)							
.	Alarm Trip Point Low (low byte) Default = Data Type minimum							
.	Alarm Trip Point Low (high byte)							
.	Alarm Hysteresis (low byte) Default = 0							
.	Alarm Hysteresis (high byte)							



Safety Analog Input Point Object, Class Code: 49<sub>Hex</sub>

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
.	Alarm Settling Time (low byte) Default = 0							
.	Alarm Settling Time (high byte)							
.	Warning Enable Default = 0							
.	Warning Trip Point High (low byte) Default = Data Type maximum							
.	Warning Trip Point High (high byte)							
.	Warning Trip Point Low (low byte) Default = Data Type minimum							
.	Warning Trip Point Low (high byte)							
.	Warning Hysteresis (low byte) Default = 0							
.	Warning Hysteresis (high byte)							
.	Warning Settling Time (low byte) Default = 0							
.	Warning Settling Time (high byte)							
.	Overrange (low byte) Default = Data Type maximum							
.	Overrange (high byte)							
.	Underrange (low byte) Default = Data Type minimum							
.	Underrange (high byte)							
.	Offset-A Data Type Default = C3							
.	Offset- A (low byte) Default = 0							
.	Offset- A (high byte)							
.	Gain Data Type Default = CA							
.	Gain (low byte) Default = 0							
.	Gain (high byte)							
.	Unity Gain Reference (low byte) Default = 1.0							
.	Unity Gain Reference (high byte)							
.	Offset-B (low byte) Default = 0							
.	Offset-B (high byte)							
.	Produce Trigger Delta (low byte) Default = 0							
.	Produce Trigger Delta (high byte)							
.	Subclass Default = 0							

**Important:** Insert default values for all unsupported attributes and return up to the last supported attribute.



### 5-14.5.2 Set\_Attributes\_All Request

No settable attributes currently exist at the Class level for the Safety Analog Input Point object.

At the Instance level, the order of attributes passed in the Set\_Attributes\_All request is as follows:

**Table 5-14.10 Set\_Attributes\_All Request Data – Instance Level**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Value Data Type (1)							
1	Input Range (3)							
2	Input Channel Mode (4)							
3	Real Time Sample Period (low byte) (7)							
4	Real Time Sample Period (high byte)							
5	Input Error Latch Time (low byte) (8)							
6	Input Error Latch Time (high byte)							
7	Full Scale (low byte) (9)							
.								
.	Full Scale (high byte)							
.	Safe State Behavior (10)							
.	Safe State Value (low byte) (11)							
.	Safe State Value (high byte)							
.	Low Signal (low byte) (12)							
.	Low Signal (high byte)							
.	High Signal (low byte) (13)							
.	High Signal (high byte)							
.	Low Engineering (low byte) (14)							
.	Low Engineering (high byte)							
.	High Engineering (low byte) (15)							
.	High Engineering (high byte)							
.	Alarm Enable (17)							
.	Alarm Trip Point High (low byte) (18)							
.	Alarm Trip Point High (high byte)							
.	Alarm Trip Point Low (low byte) (19)							
.	Alarm Trip Point Low (high byte)							
.	Alarm Hysteresis (low byte) (20)							
.	Alarm Hysteresis (high byte)							
.	Alarm Settling Time (low byte) (21)							
.	Alarm Settling Time (high byte)							
.	Warning Enable (22)							
.	Warning Trip Point High (low byte) (23)							
.	Warning Trip Point High (high byte)							
.	Warning Trip Point Low (low byte) (24)							
.	Warning Trip Point Low (high byte)							
.	Warning Hysteresis (low byte) (25)							



Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Warning Hysteresis (high byte)							
	Warning Settling Time (low byte) (26)							
	Warning Settling Time (high byte)							
	Offset- A (low byte) (30)							
	Offset- A (high byte)							
	Gain (low byte) (32)							
	Gain (high byte)							
	Offset-B (low byte) (34)							
	Offset B (high byte)							
	Produce Trigger Delta (low byte) (35)							
	Produce Trigger Delta (high byte)							

**Important:** The Set\_Attributes\_All service is to be supported only if all settable attributes shown above are implemented as settable.

## 5-14.6 Object-specific Services

The Safety Analog Input Object provides the following Object Specific Services:

**Table 5-14.11 Safety Analog Input Point Object Specific Services**

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
4Bhex	n/a	Optional	Zero_Adjust	Causes the module to modify the offset value used for calculating the Safety Analog Input Value.
4Chex	n/a	Optional	Gain_Adjust	Causes the module to modify the gain value used for calculating the Safety Analog Input Value

**Important:** Services are only accepted when the Safety Supervisor is in the Configuring or Waiting for TUNID state.

### 5-14.6.1 Zero\_Adjust and Gain\_Adjust Services

The Zero\_Adjust and Gain\_Adjust services are used to cause the Safety Analog Input to update the offset and gain values based on manufacturer specific algorithms. The target value specified in the service request represents the actual parametric measurement that the physical sensor should be reporting at the time of the calibration request.

**Table 5-14.12 Zero\_Adjust Request Service Data Parameters**

Parameter	Required	Data Type	Description	Semantics of Values
Target Value	Optional	Specified by <i>Data Type</i> attribute	Target value for the zero calibration	The value to which the <i>Safety Analog Input Value</i> attributes will be set. Default = 0



**Table 5-14.13 Gain Adjust Request Service Data Parameters**

Parameter	Required	Data Type	Description	Semantics of Values
Target Value	Required	Specified by <i>Data Type</i> attribute	Target value for the zero calibration	The value to which the <i>Safety Analog Input Value</i> attribute will be set

When the requested service has been completed successfully, a success response is returned. If the request fails, one of the following error responses is returned.

**Table 5-14.14 Error Responses**

General Status	Extended Status	Description
10	na	Device State Conflict, can not accept request in current state
20	01	Calibration Failure, unable to apply the specified values

## 5-14.7 Behavior

This section defines the required behaviors of the Safety Analog Input Point.

### 5-14.7.1 Type of Safety Inputs

The following table defines the type of safety inputs supported by the Safety Analog Input Point Object. The effects on the Raw Input Value, Analog Input Value, Safety Analog Input Value and Status are shown. The SAIP is designed to execute the selected Safe State Behavior when a safety related fault condition is detected.

**Table 5-14.15 Supported Safety Input Types**

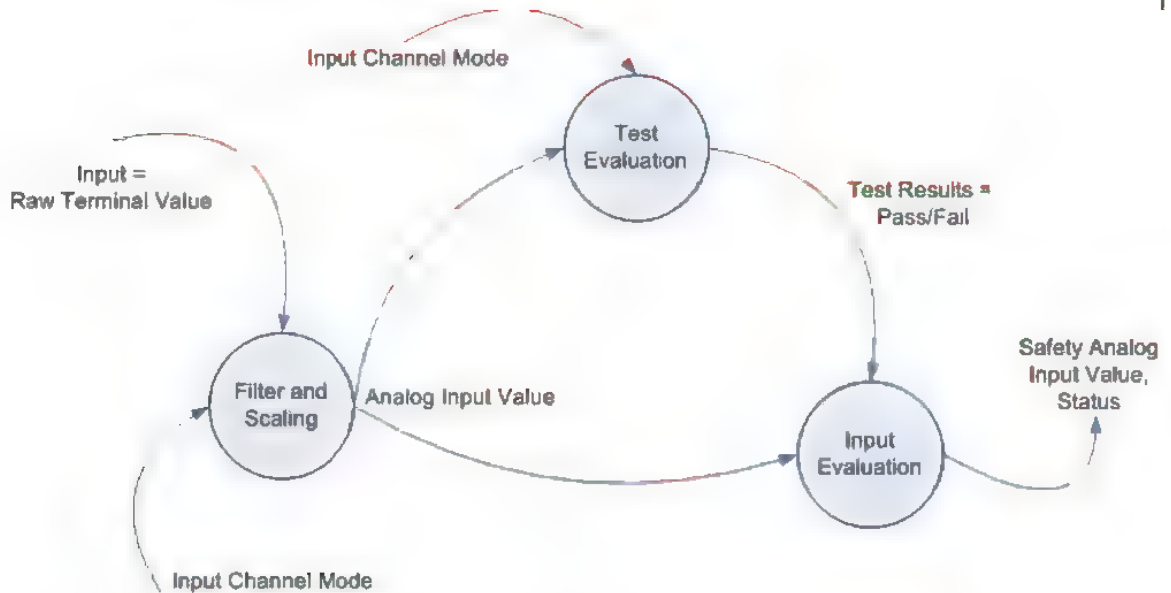
Input Channel Type	Safety Input Type	
0 = Input Signal Not Used	Raw Input Value	Undefined
	Analog Input Value	Undefined
	Safety Analog Input Value	<i>Safe State Behavior</i> Value
	Input Point Status	Always 0, = Invalid
1 = Safety, with SIL integrity testing	Raw Input Value	Input Terminal Value
	Analog Input Value	Set to resulting value after conditioning and filtering
	Safety Analog Input Value	Set to resulting value after conditioning and filtering, and SIL integrity testing. Set to <i>Safe State Value</i> if fault is detected.
	Input Point Status	0 = Faulted/Invalid 1 = Valid
2 = Standard, no SIL integrity testing	Raw Input Value	Input Terminal Value
	Analog Input Value	Set to resulting value after conditioning and filtering
	Safety Analog Input Value	Set to resulting value after conditioning and filtering
	Input Point Status	0 = Faulted/Invalid 1 = Valid



### 5-14.7.2 Safety Analog Input Evaluation

The following diagram depicts one possible flow on Safety Analog Input Evaluation processing,

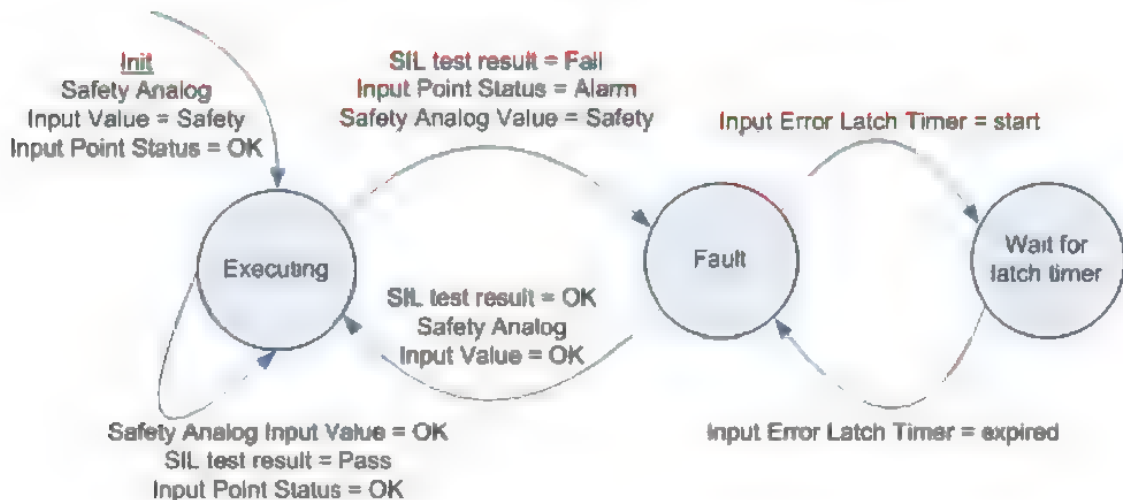
**Figure 5-14.3 Safety Analog Input Evaluation Flow**



The details of the test evaluation are product specific but generally include verification of signal integrity, validity and accuracy. Test may also cover signal path connectivity, strength or other transmission characteristics. The Safety Analog Input Value shall be set according to Safe State Behavior attribute when the Input channel mode is “Not Used”.

The following diagram defines the behavior of the Input Evaluation process shown above.

**Figure 5-14.4 Safety Analog Input Evaluation**





## 5-15 Safety Analog Input Group Object

### Class Code: 4A<sub>hex</sub>

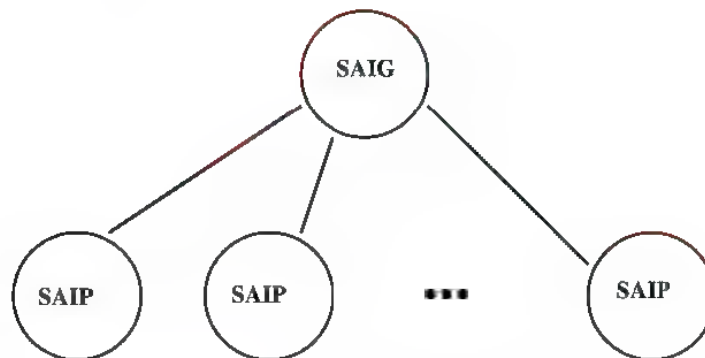
The Safety Analog Input Group Object (SAIG) binds a group of Safety Analog Input Points (SAIP) in a module. All points bound to the group share all attributes contained in the group. If an attribute is shared (e.g. if points have the same configuration attributes AND the same attribute values) across more than one Safety Analog Input Point, then that attribute can be contained in a Safety Analog Input Group. A Safety Analog Input Point can be bound to more than one Safety Analog Input Group, but the attributes that are common at the group level must have the same values in each group. Configuration attributes contained in the group are Get only in the individual SAIP object.

The list of Safety Analog Input points bound to the group is static and is a Get-only attribute of the group.

When to use the Safety Analog Input Group Object:

- When a single attribute is shared among many input points.
- To more efficiently access data (services that affect all members of the group can be supported by the SAIG).

**Figure 5-15.1 Safety Analog Input Group Object Application**



### 5-15.1 Revision History

The table below represents the revision history of this object:

**Table 5-15.1 Safety Analog Input Group Object Revision History**

Revision	Reason for object definition update
01	Initial Definition at First Release of Specification

### 5-15.2 Class Attributes

**Table 5-15.2 Safety Analog Input Group Object Class Attributes**

Attribute ID	Need in Implementation	Access Rule	Name	Data Type	Description of Attribute	Semantics of Values
1 thru 7	These class attributes are optional and are described in Chapter 4 of this specification.					



**5-15.3 Instance Attributes****Table 5-15.3 Safety Analog Input Group Object Instance Attributes**

Attr ID	Need in Implem	Access Rule	NV	Attribute Name	Data Type	Description of Attribute	Semantics of Values
1	Optional	Get	NV	Number of Bound Instances	USINT	Number of Safety Analog Input points bound to this group	
2	Optional	Get	NV	Binding	Array of UINT	List of Instance Ids of Safety Analog Input points bound to this group	
3	Optional	Set <sup>1</sup>	NV	Data Type	USINT	Determines the data type of the Safety Analog Input Value attribute for all members of this group	C3 = INT (default) See Appendix C-6 1
4	Optional	Set <sup>1</sup>	NV	Input Range	USINT	Input range the point is operating in	See Semantics section.
5	Optional	Get	V	Input Status	BOOL	Logical AND of the Input Point Status of the Analog Input Points in the Group	0 = One or more group members are faulted 1 = Data from all members is valid.
6	Optional	Set	NV	Input Error Latch Time	UNIT	Minimum time a Safety Input error will be latched for.	0 – 65535 ms, default = 1000 See Semantics section.
7	Optional	Set	NV	Full Scale	INT or based on <i>Data Type</i>	The value of Full Scale for the sensor	The value of <i>Safety Analog Input Value</i> corresponding to the Full Scale calibrated measurement of the sensor. Default = Maximum value for the <i>Data Type</i> .
8	Optional	Set <sup>1</sup>	NV	Safe State Behavior	USINT	Defines behavior for value reporting when faulted.	0 = Use Safe State Value, default. See Semantics section.
9	Optional	Set <sup>1</sup>	NV	Safe State Value	INT or based on <i>Data Type</i>	Safe State Analog Input value.	0 – default.
10	Optional	Get	V	Alarm and Warning Status	BYTE	Logical OR of all Alarm and Warning Status of the analog points in the Group.	0 = No alarms or warnings in the group 1 = Alarm or Warning in at least one group member
11	Optional	Set	NV	Alarm Enable	BOOL	Enable Alarm Status Checking	0 = Disable, default 1 = Enable
12	Optional	Set	NV	Alarm Trip Point High	INT or based on <i>Data Type</i>	Specifies the <i>Value</i> above which an Alarm Condition will occur	See Semantics section, Default = Max value for <i>Data Type</i>



Attr ID	Need in Implem	Access Rule	NV	Attribute Name	Data Type	Description of Attribute	Semantics of Values
13	Optional	Set	NV	Alarm Trip Point Low	INT or based on <i>Data Type</i>	Specifies the <i>Value</i> below which an Alarm Condition will occur.	See Semantics section, Default = Min value for <i>Data Type</i>
14	Optional	Set	NV	Alarm Hysteresis	INT or based on <i>Data Type</i>	Specifies the amount by which the Safety Analog Input Value must recover to clear the Alarm condition.	0 = default See Semantics section.
15	Optional	Set	NV	Alarm Settling Time	INT or based on <i>Data Type</i>	Determines the time that the Value must exceed the Alarm Trip Point before a Warning Condition will occur.	Time in milliseconds 0 = default See Semantics section.
16	Optional	Set	NV	Warning Enable	BOOL	Enable Alarm Status Checking	0 = Disable, default 1 = Enable
17	Optional	Set	NV	Warning Trip Point High	INT or based on <i>Data Type</i>	Specifies the <i>Value</i> above which an Alarm Condition will occur.	See Semantics section, Default = Max value for <i>Data Type</i>
18	Optional	Set	NV	Warning Trip Point Low	INT or based on <i>Data Type</i>	Specifies the <i>Value</i> below which an Alarm Condition will occur.	See Semantics section, Default = Min value for <i>Data Type</i>
19	Optional	Set	NV	Warning Hysteresis	INT or based on <i>Data Type</i>	Specifies the amount by which the Safety Analog Input Value must recover to clear the Warning condition	0 = default See Semantics section.
20	Optional	Set	NV	Warning Settling Time	INT or based on <i>Data Type</i>	Determines the time that the Value must exceed the Warning Trip Point before a Warning Condition will occur.	Time in milliseconds 0 = default See Semantics section.

1. Attribute may be implemented as get only. Refer to Safety Analog Input Point.

## 5-15.4 Semantics

This section defines the semantics of the Safety Analog Input Group object attributes.

### 5-15.4.1 Number of Bound Instances, Attribute 1

Number of Safety Analog Input points bound to the group.

### 5-15.4.2 Bound Instances, Attribute 2

List of instance Ids of Safety Analog Input points bound to the group.



**5-15.4.3 Data Type, Attribute 3**

The Data Type attribute determines the data type to be used by all members of the group. Refer to the Safety Analog Input Point object.

**5-15.4.4 Input Range, Attribute 4**

The Input Range attribute determines the set of analog values within which the inputs may operate. The value of the Input Range affects the default and legal values that other attributes may have. Refer to the Safety Analog Input Point object.

**5-15.4.5 Input Status, Attribute 5**

Status is a simple logical AND of the Input Point Status of each SAIP instance bound to the Safety Analog Input Group.

**5-15.4.6 Input Error Latch Time, Attribute 6**

The Input Error Latch Time specifies the minimum time that an input point fault shall be latched after the fault condition is detected for all members of the group. Refer to the Safety Analog Input Point object.

**5-15.4.7 Full Scale, Attribute 7**

Value returned to the controller when the Safe State Behavior attribute is set to “Full Scale” (1). The Full Scale Value is determined by the Data Type and Input Range. Default value is the maximum value of the data type.

**5-15.4.8 Safe State Behavior, Attribute 8**

The Safe State Behavior attribute specifies for all members of the group the action that the Safety Analog Input Point should report to the safety controller whenever the input channel is in the “Safe State”, e.g. a fault in the input or power circuitry is detected. Refer to the Safety Analog Input Point object.

**5-15.4.9 Safe State Value, Attribute 9**

The Safe State Value attribute specifies for all members of the group the analog input value that is reported to the safety controller whenever the Input Point Status is 0 and the Safe State Behavior is Use Safe State Value. The default is zero.

**5-15.4.10 Alarm and Warning Status, Attribute 10**

Status is a simple logical OR of the Alarm and Warning Status of each SAIP instance bound to the Safety Analog Input Group.



#### 5-15.4.11 Alarm/Warning Enable, Attributes 11, 16

The Alarm and Warning Enable attributes allow the user to specify when the process alarms and/or warnings are to be monitored. When set to 1, the alarms and warnings are monitored and the associated status bits in the Alarm and Warning Status attribute are set to one or zero, as appropriate depending on the alarm or warning trip point, hysteresis and settling time attributes.

#### 5-15.4.12 Alarm/Warning Trip Points, Hysteresis and Settling Time, Attributes 12-15, 17-20

The Alarm or Warning Trip Point High attribute is the level above which an alarm or warning will be declared when the Safety Analog Input Value exceeds this level.

The Alarm or Warning Trip Point Low attribute is the level below which an alarm or warning will be declared when the Safety Analog Input Value is less than this level.

Before the Alarm and Warning configuration is applied, the values shall be validated to ensure that:

- Alarm High Trip Point  $\geq$  Warning Trip Point High
- Alarm High Trip Low  $\leq$  Warning Trip Point Low
- Alarm High Trip Point  $>$  Alarm High Trip Low
- Warning Trip Point High  $>$  Warning Trip Point Low

The Alarm or Warning Hysteresis attribute specifies the amount by which the Safety Analog Input Value for all members in the group must transition in order to clear an alarm or warning condition. Refer to the Safety Analog Input Point object.

The Alarm or Warning Settling Time attribute specifies the amount of time that the Safety Analog Input Value attribute for all members in the group must exceed the Alarm or Warning Trip Point level before an alarm or warning is declared. The Alarm or Warning Settling Time also applies to the clearing of the alarm or warning.

### 5-15.5 Common Services

The Safety Analog Input Group Object provides the following Common Services:

**Table 5-15.4 Safety Analog Input Group Object Common Services**

Service Code	Need In Implementation		Service Name	Description of Service
	Class	Instance		
0E hex	Conditional <sup>1</sup>	Required	Get_Attribute_Single	Returns the contents of the specified attribute
10 hex	N/A	Conditional <sup>2</sup>	Set_Attribute_Single	Modifies an attribute value
01 hex	Optional	Optional	Get_Attributes_All	Returns a predefined listing of this object's attributes (See the Get_Attributes_All Response definition below)
02 hex	N/A	Optional	Set_Attributes_All	Modifies the contents of the attributes of the class or object.

1. The Get\_Attribute\_Single service is REQUIRED if any attributes are implemented

2. The Set\_Attribute\_Single service is required only if any Settable Attributes are implemented

See Appendix A for definitions of these services.



### 5-15.5.1 Get\_Attributes\_All Response

At the Class level, the order of the attributes returned in the “Object/service specific reply data” portion of the Get\_Attributes\_All response are defined in Volume 1, Chapter 4.

**Important:** Insert default values for all unsupported attributes.

At the Instance level, the order of the attributes returned in the “Object/service specific reply data” portion of the Get\_Attributes\_All response is as follows:

**Table 5-15.5 Get\_Attributes\_All Response Data – Instance Level**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Number of Bound Instances (default = 0)							
1	Binding : Instance ID #1 (low byte)							
2	Binding : Instance ID #1 (high byte)							
...	Binding : Instance ID #m (low byte)							
.	Binding : Instance ID #m (high byte)							
...	Data Type (default = C3)							
.	Input Range Default = 3							
	0	0	0	0	0	0	0	Input Status Default = 1
.	Input Error Latch Time (low byte) (default = 0)							
	Input Error Latch Time (high byte)							
.	Full Scale (low byte) Default = Data Type Maximum							
	Full Scale (high byte) Default = Data Type Maximum							
	Safe State Behavior =(default = 0)							
	Safe State Value (low byte) (default = 0)							
.	Safe State Value (high byte) (default = 0)							
	Alarm/Warning Status (default = 0)							
.	Alarm Enable Default = 0							
	Alarm Trip Point High (low byte) Default = Data Type maximum							
.	Alarm Trip Point High (high byte)							
	Alarm Trip Point Low (low byte) Default – Data Type minimum							
.	Alarm Trip Point Low (high byte)							
	Alarm Hysteresis (low byte) (default = 0)							
.	Alarm Hysteresis (high byte) (default = 0)							
	Alarm Settling Time (low byte) (default = 0)							
.	Alarm Settling Time (high byte) (default = 0)							
	Warning Enable Default = 0							
.	Warning Trip Point High (low byte) Default = Data Type maximum							
	Warning Trip Point High (high byte)							
.	Warning Trip Point Low (low byte) Default = Data Type minimum							
	Warning Trip Point Low (high byte)							



Safety Analog Input Group Object, Class Code: 4A<sub>Hex</sub>

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
.	Warning Hysteresis (low byte) (default = 0)							
.	Warning Hysteresis (high byte) (default = 0)							
.								
.								
.	Warning Settling Time (low byte) (default = 0)							
.	Warning Settling Time (high byte) (default = 0)							

**Important:** Insert default values for all unsupported attributes and return up to the last supported attribute.

**Important:** If the instance attribute "Number of Bound Instances" is not supported, the default value of zero is to be inserted into the response byte array and no Binding data will follow.

### 5-15.5.2 Set\_Attributes\_All Request

No settable attributes currently exist at the Class level for the Safety Analog Input Group Object.

At the Instance level, the order of attributes passed in the "Object/service specific request data" portion of the Set\_Attributes\_All request is as follows:

**Table 5-15.6 Set\_Attributes\_All Request Service Data – Instance Level**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Data Type (3)							
1	Input Range Default = 3							
2	Input Error Latch Time (low byte)							
3	Input Error Latch Time (high byte)							
4	Full Scale (low byte) Default = Data Type Maximum							
.	Full Scale (high byte) Default = Data Type Maximum							
.	Safe State Behavior							
.	Safe State Value (low byte)							
.	Safe State Value (high byte)							
.	Alarm Enable							
.	Alarm Trip Point High (low byte)							
.	Alarm Trip Point High (high byte)							
.	Alarm Trip Point Low (low byte)							
.	Alarm Trip Point Low (high byte)							
.	Alarm Settling Time (low byte)							
.	Alarm Settling Time (high byte)							
.	Alarm Hysteresis (low byte)							
.	Alarm Hysteresis (high byte)							
.	Warning Hysteresis (low byte)							
.	Warning Hysteresis (high byte)							
.								
.								



Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
.	Warning Enable							
.	Warning Trip Point High (low byte)							
.	Warning Trip Point High (high byte)							
.	Warning Trip Point Low (low byte)							
.	Warning Trip Point Low (high byte)							
.	Warning Settling Time (low byte)							
.	Warning Settling Time (high byte)							

**Important:** The Set\_Attributes\_All service is to be supported only if all settable attributes shown above are implemented as settable.

### 5-15.6 Object-specific Services

The Safety Analog Input Group Object provides no Object-specific services.

### 5-15.7 Behavior

The primary purpose of the Safety Analog Input Group Object is to bind Safety Analog Input Points.

An attribute of the SAIG modifies the behavior or reports additional status information of all SAIPs bound to the group.

The State Transition Diagram below provides a graphical description of the events and corresponding state transitions. A subset of the states and events may be supported in an application, but the behavior must be consistent.

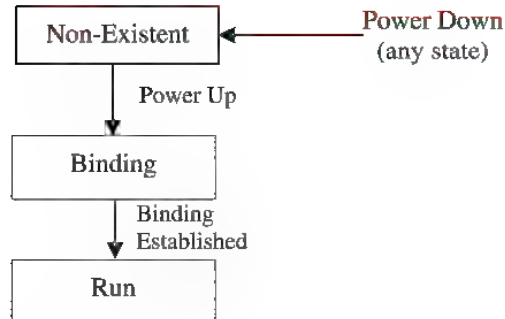
The states shown are equivalent to the following:

**Table 5-15.7 Safety Analog Input Group Object State Definitions**

This state	Is equivalent to
Non-Existent	a module without power
	a module that has an unrecoverable fault (e.g. processor watchdog timeout)
	a major reconfiguration of the module (e.g. NVS Update)
Binding	when analog input points are added or bound to each predefined Safety Analog Input Group instance (executed as part of Power Up cycle)
Run	the point at which Get and Set attribute services can be used to access the attributes of the Group Object



**Figure 5-15.2 State Transition Diagram for Safety Analog Input Group Object**



#### **5-15.7.1 Attribute Access Rules**

All attributes are gettable or settable according to their attribute access rules. The Settable attributes are only settable when the Safety Supervisor is in the Configuring state.

#### **5-15.7.2 Input Status Attribute**

The Input Status attribute is simply a logical AND of each individual Input Point Status for all points bound to the group.

#### **5-15.7.3 Alarm and Warning Status Attribute**

The Alarm and Warning Status attribute is simply a logical OR of each individual Alarm and Warning Status for all process failures or alarm conditions for all points bound to the group.



## 5-16 Safety Dual Channel Analog Input Object

### Class Code: 4B<sub>hex</sub>

The Safety Dual Channel Analog Input (SDCAI) Object models a pair of safety analog input channels as Safety Dual Channel Analog Inputs in a product. This object must be used in conjunction with two instances of a Safety Analog Input Point Object. This object can be used in any device that contains two or more safety analog inputs. Note that the term “input” is defined from the network’s point of view. An input will produce data on the network.

The main feature of the Safety Dual Channel Analog Input is that it allows a pair of diverse analog input values to be monitored simultaneously. This provides signal level integrity for the resulting input by running various SIL integrity tests on the input signals before reporting the input values to the control system. Whenever a discrepancy fault is detected, a fault state is declared and a discrepancy value is reported to the control system.

### 5-16.1 Revision History

This is the initial release of this object. The table below represents the revision history:

**Table 5-16.1 Safety Dual Channel Analog Input Object Revision History**

Revision	Reason for object definition update
01	Initial Definition at First Release of Specification. CIP Safety 2.3

### 5-16.2 Class Attributes

**Table 5-16.2 Safety Dual Channel Analog Input Object Class Attributes**

Attr ID	Need in Implem	Access Rule	NV	Name	Data Type	Description of Attribute	Semantics of Value
1 thru 7	See the Volume 1, Chapter 4-9.1 for a description of the optional, reserved class attributes.						

### 5-16.3 Instance Attributes

**Table 5-16.3 Safety Dual Channel Analog Input Object Instance Attributes**

Attr ID	Need in Implem	Access Rule	NV	Attribute Name	Data Type	Description of Attribute	Semantics of Values
1	Required	Set	NV	Dual Channel Mode	USINT	Selects the mode for the two channels of the Safety Dual Channel Analog Input.	0: Single Channel (default) 1: Dual Channel Equivalent 2: – 255: Reserved
2	Optional	Get	V	Dual Channel Evaluation Status	BOOL	Status of the Dual Channel evaluation	0: Discrepancy Detected 1: Valid This is Safety Data, Safety State Value = 0



Safety Dual Channel Analog Input Object, Class Code: 4B<sub>Hex</sub>

Attr ID	Need in Implem	Access Rule	NV	Attribute Name	Data Type	Description of Attribute	Semantics of Values
3	Required	Set <sup>1</sup>	NV	Discrepancy Timer	UINT	The time limit at which the input discrepancy becomes an error.	Range: 0 to 65565, in milliseconds Default = 0, no monitoring
4	Required	Set <sup>1</sup>	NV	Member One	USINT	Instance Id of one of the pair of Safety Analog Input Point (SAIP) objects that forms the Safety Dual Channel Input. (Channel A)	Default = 0, no pairing
5	Required	Set <sup>1</sup>	NV	Member Two	USINT	Second member of the dual channel safety input pair. (Channel B)	Default = 0, no pairing
6	Required	Set <sup>1</sup>	NV	Discrepancy Deadband	INT or based on SAIP Data Type	Allowed difference for the channel A and B values	Default = 0, no Deadband See Semantics
7	Optional	Set <sup>1</sup>	NV	Discrepancy Behavior	USINT	Defines behavior for value reporting when discrepancy occurs.	3 – Continue Sensing, default. See Semantics section.
8	Optional	Set <sup>1</sup>	NV	Discrepancy Value	INT or based on Data Type	Analog Input value when discrepancy occurs	0 = default.

1. May be implemented as Get only

## 5-16.4 Semantics

This section defines the semantics of the Safety Dual Channel Analog Input object attributes.

### 5-16.4.1 Dual Channel Mode, Attribute 1

The *Dual Channel Mode* attribute specifies how the analog input signal is used in the application.

0 – Single. No dual Channel Evaluation is applied.

1 – Dual Channel Equivalent. The primary input signal is compared to the secondary input signal and evaluated for equality as determined by the Discrepancy Deadband and Discrepancy Timer.



#### **5-16.4.2 Dual Channel Evaluation Status, Attribute 2**

The *Dual Channel Evaluation Status* attribute provides a status of the Safety Dual Channel Analog Input. When set to 1, the Input Point is operating normally. When set to 0, a discrepancy fault has been declared. The *Dual Channel Evaluation Status* is associated with the bound SAIP *Input Point Status*. When a discrepancy fault is declared, the associated *Input Point Status* is set to 0 for both Member 1 and Member 2. The *Point Fault Reason* is set to Discrepancy Error (5) for Member 1 and Partner Discrepancy Error (6) for Member 2.

#### **5-16.4.3 Discrepancy Timer, Attribute 3**

The *Discrepancy Timer* attribute defines how the discrepancy time monitoring is applied. If *Discrepancy Timer* is set to zero, no discrepancy time is applied. If non-zero, no discrepancy fault is declared until after the discrepancy timer has expired. The discrepancy fault is cleared when the signals return to an acceptable *Discrepancy Deadband* value for the *Discrepancy Timer* time. The *Discrepancy Behavior* value will be returned to the control system when the discrepancy exists for longer than the *Discrepancy Timer* time.

#### **5-16.4.4 Member One, Attribute 4**

The *Member One* attribute defines the first instance of the Safety Analog Input Pair. If a discrepancy fault is declared, this input shall be assigned the Discrepancy Error in the *Fault Reason* attribute of the SAIP.

When used in the Dual mode, both members of the pair must have similar configurations for the following attributes of the underlying SAIP instances. How this is accomplished is vendor specific.

1. Data Type (1)
2. Input Range (3)
3. Input Channel Mode (4)
4. Real Time Sample Period (7)

#### **5-16.4.5 Member Two, Attribute 5**

The *Member Two* attribute defines the second instance of the Safety Analog Input Pair. If a discrepancy fault is declared, this input shall be assigned the Partner Discrepancy Error in the *Fault Reason* attribute of the SAIP.

#### **5-16.4.6 Discrepancy Deadband, Attribute 6**

The *Discrepancy Deadband* attribute defines a difference value to be used when evaluating the two paired analog input signals. The value is an absolute value of the maximum difference to be accepted when the two Safety Analog Input Values are compared. If a discrepancy occurs, the *Discrepancy Timer* time will be applied before the safe state data value is returned in the Safety Analog Input Values for the Dual Channel Input pair.



**5-16.4.7 Discrepancy Behavior, Attribute 7**

The *Discrepancy Behavior* attribute specifies the value that the Safety Analog Input Points in the Dual Channel pair should report in the *Safety Analog Input Value* to the safety controller whenever a discrepancy condition is detected.

**Table 5-16.4 Discrepancy Behavior Definitions**

Value	Definition
0	Use Discrepancy Value
1	Full Scale
2	Hold Last Value
3	Continue Sensing (Default)
4-99	Reserved
100-255	Vendor Specific

**5-16.4.8 Discrepancy Value, Attribute 8**

The *Discrepancy Value* attribute represents the analog input value that is reported in the *Safety Analog Input Value* to the safety controller whenever a discrepancy condition is detected and the *Discrepancy Behavior* is Use Discrepancy Value. The default is zero.

**5-16.5 Common Services**

The Safety Dual Channel Analog Input Object provides the following Common Services:

**Table 5-16.5 Safety Dual Channel Analog Input Object Common Services**

Service Code	Need in Implementation		Service Name	Description of Service
	Class	Instance		
0Ehex	Conditional <sup>1</sup>	Required	Get_Attribute_Single	Returns the contents of the specified attribute.
01hex	Optional	Optional	Get_Attributes_All	Returns a predefined listing of this objects attributes (See the Get_Attributes_All Response definition below)
02hex	n/a	Optional	Set_Attributes_All	Modifies the contents of the attributes of the class or object.

1. Required if any attributes are supported.

See Appendix A for definition of these services

**5-16.5.1 Get\_Attributes\_All Response**

At the Class level, the order of the attributes returned in the “Object/service specific reply data” portion of the Get\_Attributes\_All response is defined in the CIP Specification, Volume 1, Chapter 4-5.1.

At the Instance level, the order of the attributes returned in the “Object/service specific reply data” portion (see Chapter 4 for a description of the Service Data field) of the Get\_Attributes\_All response is as follows:



**Table 5-16.6 Get Attributes All Response Data – Instance Level**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Dual Channel Mode Default = 0							
1	Dual Channel Evaluation Status Default = 0							
2	Discrepancy Timer (high byte) Default = 0							
3	Discrepancy Timer (low byte)							
4	Member One Default = 0							
5	Member Two Default = 0							
6	Discrepancy Deadband (high byte) Default = 0							
n	Discrepancy Deadband (low byte)							
n+1	Discrepancy Behavior Default = 3							
n+2	Discrepancy Value (high byte) Default = 0							
n+3	Discrepancy Value (low byte)							

**Important:** Insert default values for all unsupported attributes.

### 5-16.5.2 Set\_Attributes\_All Request

No settable attributes currently exist at the Class level for the Safety Dual Channel Analog Input object.

At the Instance level, the order of attributes passed in the Set\_Attributes\_All request is as follows:

**Table 5-16.7 Set Attributes All Request Data – Instance Level**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Dual Channel Mode (1)							
1	Discrepancy Timer (high byte) (3)							
2	Discrepancy Timer (low byte)							
3	Member One (4)							
4	Member Two (5)							
5	Discrepancy Deadband (high byte) (6)							
n	Discrepancy Deadband (low byte)							
n+1	Discrepancy Behavior (7)							
n+2	Discrepancy Value (high byte) (8)							
n+3	Discrepancy Value (low byte)							

**Important:** The Set Attributes All service is to be supported only if all settable attributes shown above are implemented as settable.

### 5-16.6 Object-specific Services

The Safety Dual Channel Analog Input Object provides no Object-specific services.



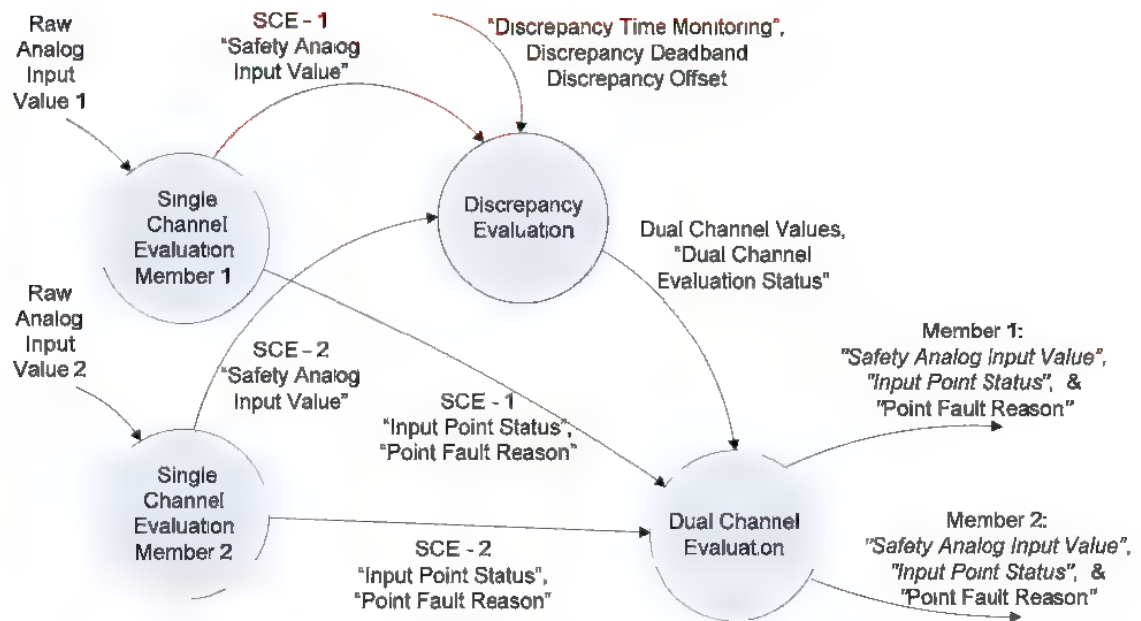
## **5-16.7 Behavior**

### **5-16.7.1 Safety Dual Channel Input Evaluation**

When instance attribute “Dual Channel Mode” (Attribute 1) is set to single, the Safety Dual Channel Analog Input object instance has no behavior beyond the instances of the Safety Analog Input Point object.

When instance attribute “Dual Channel Mode” is set to equivalent, there is an interaction between the SDCAI object instance and the two SAIP member instances. The following diagram illustrates this relationship. It depicts the flow of the Dual Channel Evaluation behavior when the “Dual Channel Mode” instance attribute is set to Equivalent.

**Figure 5-16.1 Overview of Dual Channel Evaluation Behavior**



The Dual Channel Evaluation Behavior is described in terms of the following:

1. Single channel evaluation of channels 1 and 2
2. Discrepancy Evaluation
3. Dual Channel Evaluation

The Single Channel Evaluation Behavior is defined within the Safety Analog Input Point object for both members used in the Safety Dual Channel Analog Input object instance.



### **5-16.7.2 Discrepancy Evaluation**

From a behavioral perspective, the Discrepancy Evaluation portion of the SDCAI object takes the following inputs:

1. Safety Analog Input Value attribute of Member 1 - Single Channel Evaluation
2. Safety Analog Input Value attribute of Member 2 - Single Channel Evaluation
3. Discrepancy Deadband attribute of the SDCAI object
4. Discrepancy Timer attribute of the SDCAI object (Time)

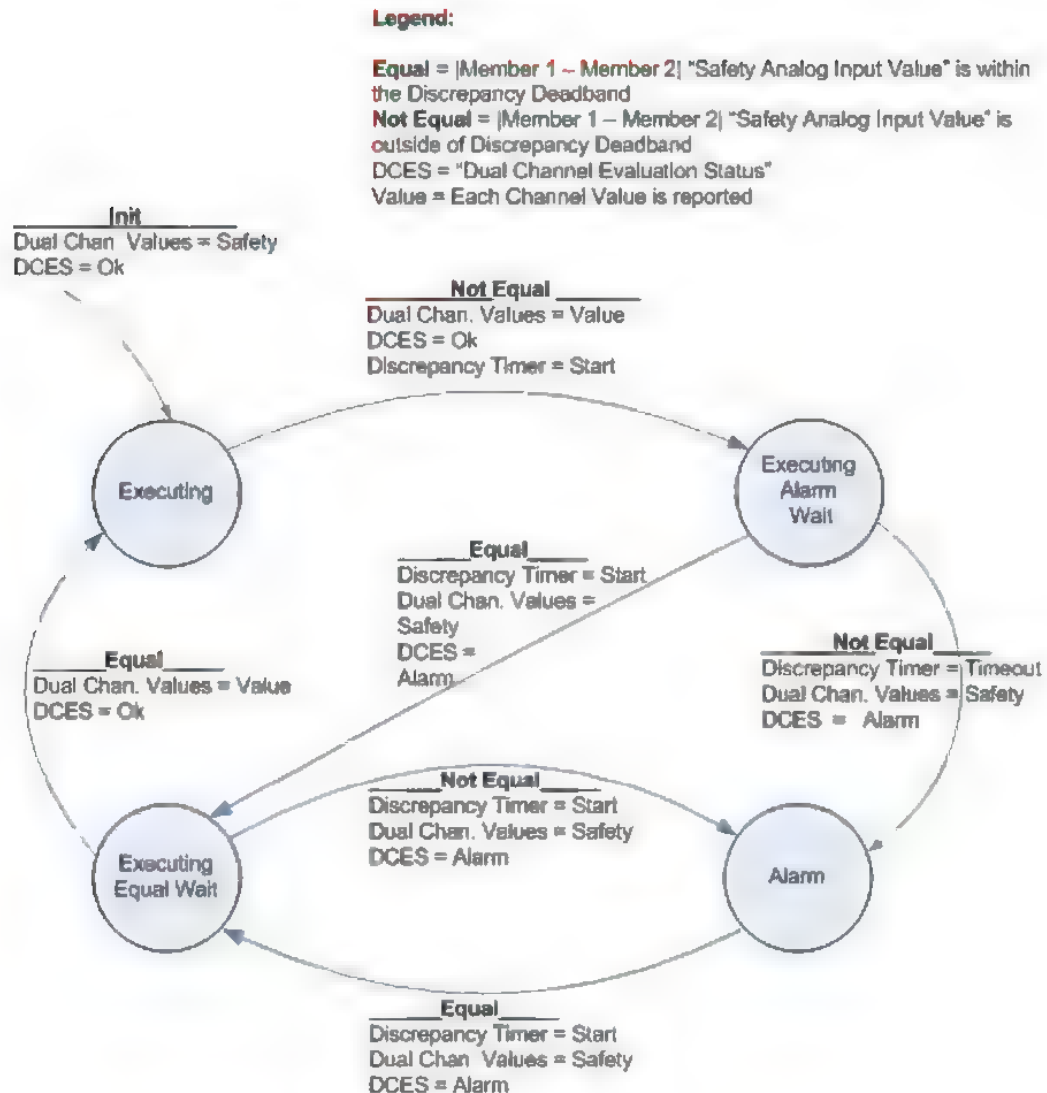
From a behavioral perspective, the Discrepancy Evaluation produces the following outputs:

1. Dual Channel Values – When the respective Safety Analog Input values are within the specified Discrepancy Deadband, the input values are reported in the Safety Analog Input Values for the Dual Channel Input pair. If a discrepancy error is detected, the input values reported are determined by the setting of the *Discrepancy Behavior* attribute of the SAIP. Both members of the dual channel pair shall have the same behavior. Refer to Figure 5-11.7-3 for further clarification.
2. *Dual Channel Evaluation Status* instance attribute of the SDCAI object (Ok or Alarm)

The following diagram illustrates the Discrepancy Evaluation behavior for Dual Channel Modes when set to Equivalent.



**Figure 5-16.2 Dual Channel Mode = Equivalent**



### 5-16.7.3 Dual Channel Evaluation

From a behavioral perspective, the Dual Channel Evaluation portion of the SDCAI object takes the following inputs:

1. Input Point Status attribute of Member 1 - Single Channel Evaluation
2. Input Point Status attribute of Member 2 - Single Channel Evaluation
3. Point Fault Reason attribute of Member 1 - Single Channel Evaluation
4. Point Fault Reason attribute of Member 2 - Single Channel Evaluation
5. Dual Channel Values that resulted from the Discrepancy Evaluation
6. Dual Channel Evaluation Status instance attribute of the SDCAI object that resulted from the Discrepancy Evaluation (Ok or Alarm)



**Safety Dual Channel Analog Input Object, Class Code: 4B<sub>Hex</sub>**

From a behavioral perspective, the Dual Channel Evaluation produces the following outputs:

1. The Safety Input Point Value, Input Point Status, and Point Fault Reason attributes for Member 1 and Member 2 Safety Analog Input Point instances.

The following table illustrates the Dual Channel Evaluation behavior for Dual Channel Modes when set to Equivalent.

**Table 5-16.8 Dual Channel Evaluation Behavior**

Single Channel Evaluation		Discrepancy Evaluation		Results applied to both Member 1 and Member 2 attributes: "Safety Analog Input Value", "Input Point Status", and "Point Fault Reason"			
Member 1 Input Point Status	Member2 Input Point Status	Dual Channel Values	Dual Channel Evaluation Status	Member 1 & Member 2 Safety Analog Input Value	Member 1 & Member 2 Input Point Status	Member1 Point Fault Reason	Member2 Point Fault Reason
Ok	Ok	Normal	Ok	Normal	OK	0x00	0x00
Alarm	Ok	x	x	Discrepancy Value	Alarm	SCE-1 Point Fault Reason	0x05
Ok	Alarm	x	x	Discrepancy Value	Alarm	0x05	SCE-2 Point Fault Reason
Alarm	Alarm	x	x	Discrepancy Value	Alarm	SCE-1 Point Fault Reason	SCE-2 Point Fault Reason
Ok	Ok	Discrepancy	Alarm <sup>1</sup>	Discrepancy Value	Alarm	0x05	0x06

x = don't care

SCE-1 = Single Channel Evaluation- Member 1

SCE-2 = Single Channel Evaluation- Member 2

Normal = Both Channels reporting normally

1. If Discrepancy Timer is not expired, Dual Channel value is Normal. Otherwise Dual Channel Value is Safe.



This page has been intentionally left blank



## **Volume 5: CIP Safety**

# **Chapter 6: Device Profiles**

---



## Contents

6-1	Safety Device Profiles .....	3
6-2	Safety State Definition.....	3
6-2.1	Digital I/O .....	3
6-2.2	Analog I/O .....	3
6-3	Baseline Safety Device .....	4
6-4	Rules for Safety Profiles .....	5
6-5	Safety Manual Requirements.....	5
6-6	Safety Discrete I/O Device .....	6
6-6.1	Object Model .....	6
6-6.2	How Objects Affect Behavior.....	8
6-6.2.1	Safety Supervisor Requirements.....	8
6-6.3	Defining Object Interfaces .....	9
6-6.4	Relationship between DIP and SDIP .....	10
6-6.5	Relationship between DOP and SDOP .....	10
6-6.6	I/O Assembly Instances.....	10
6-6.6.1	Standard Data Only Open Assemblies.....	12
6-6.6.2	Safety Data Only Open Assemblies .....	13
6-6.6.3	Safety Data and Standard Data Open Assemblies .....	15
6-6.7	I/O Assembly Data Attribute Format .....	17
6-6.7.1	Standard Data Only Open Assemblies... ..	17
6-6.7.2	Safety Data Only Open Assemblies .....	18
6-6.7.3	Safety Data and Standard Data Open Assemblies .....	30
6-6.8	Mapping I/O Assembly Data Attribute Components .....	41
6-6.8.1	Standard Data Only Open Assemblies... ..	41
6-6.8.2	Safety Data Only Open Assemblies .....	41
6-6.8.3	Safety Data and Standard Data Open Assemblies .....	43
6-7	Safety Analog I/O Device.....	45
6-7.1	Object Model .....	45
6-7.2	How Objects Affect Behavior.....	47
6-7.2.1	Safety Supervisor Requirements.....	47
6-7.3	Defining Object Interfaces .....	48
6-7.4	Relationship between AIP and SAIP .....	49
6-7.5	Relationship between AOP and SAOP .....	49
6-7.6	I/O Assembly Instances.....	49
6-7.6.1	Safety and Standard Data Open Assemblies.....	50
6-7.7	I/O Assembly Data Attribute Format ....	62
6-7.7.1	Safety and Standard Data Open Assemblies.....	62
6-7.8	Mapping I/O Assembly Data Attribute Components .....	89



## **6-1 Safety Device Profiles**

This chapter contains the Safety Device profiles.

## **6-2 Safety State Definition**

In safety systems, it is necessary to define the safety states that are present in the system. When using the CIP Safety Networks the common safety states will be defined in this section unless otherwise noted.

### **6-2.1 Digital I/O**

SRS48 The default safety state for digital inputs and outputs shall be off.

This means that digital inputs in the safety state shall report an off or zero condition along with fault status indicating that devices are in the safety state. Digital outputs that are in a safety state shall be set to off or zero volts. Digital Outputs shall also report a safety condition on their diagnostic information.

An optional state for digital inputs and outputs may be provided. This functionality should allow the affected inputs outputs to maintain last state. This option should be user configurable and the default set by the manufacturer shall be the off or zero state.

### **6-2.2 Analog I/O**

The default for both analog inputs and outputs should be the value that corresponds to a zero state or the lowest value within the range of the device. This value should represent a zero to the user.

An optional state for analog inputs and outputs may be provided.

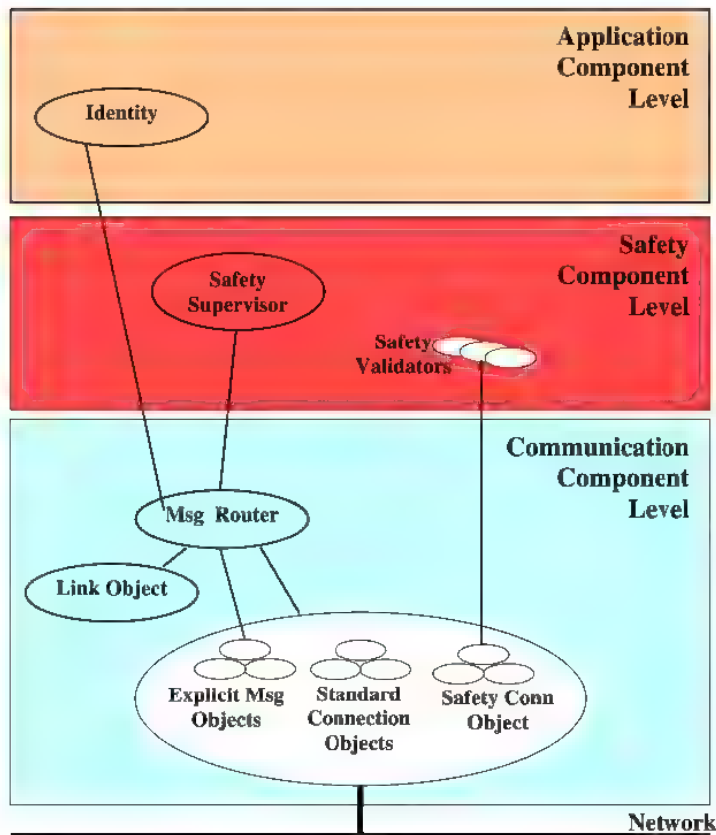


### 6-3 Baseline Safety Device

All Safety Device Profile definitions shall contain, at a minimum, a basic level of safety functionality. A complete safety profile definition is built off of this baseline. The baseline safety functionality is defined as one which contains a “baseline” Safety Supervisor (refer to Section 5-1.3) for high-integrity device control and one or more Safety Validator instances for high-integrity safety I/O connections.

It is optional for baseline safety profiles to support the SNCT implementation of the Safety Supervisor, but is highly recommended since the profile then includes a common, TUV-certified configuration solution. All safety profiles shall define which level (i.e. Baseline or SNCT) of Safety Supervisor is required.

Figure 6-3.1 Baseline Safety Device





## 6-4 Rules for Safety Profiles

If the safety profiles defined here are not suitable for a device, the device manufacturer shall define a vendor specific profile and extended device type.

Safety objects cannot be added to an existing standard profile and use that existing profile device type; a new device profile shall be created for safety.

All safety device profiles created shall have the word “Safety” as the first word in the name.

## 6-5 Safety Manual Requirements

This section will contain some of the specific requirements that must be contained in the user safety manual for devices which use the CIP Safety Network Protocol. These requirements are derived from the safety case where the user is required to perform some action to insure a safe process.

**Table 6-5.1 Safety Manual Requirements**

User Manual Requirements
SRS50 The safety manual shall contain a user instruction requiring the user to completely test a device’s operation before setting the Lock Attribute
SRS51 The safety manual shall contain a user instruction requiring the user to upload and compare the configuration from each affected safety devices to that which was sent by the SNCT before setting the Lock Attribute in those devices.
SRS52 The safety manual shall contain a user instruction requiring the user to clear any pre-existing configuration from any safety device before installing it onto a safety network.
SRS53 The safety manual shall contain a user instruction requiring the user to commission all safety devices with MacId (and Baud Rate if necessary) prior to installing it onto a safety network
SRS193 The Safety manual shall contain a user warning advising that originators that have an “automatic” SNN setting feature should only use that feature when the safety system is not being relied upon.



## 6-6 Safety Discrete I/O Device

### Device Type: 23<sub>hex</sub>

A Safety Discrete I/O Device type interfaces to multiple Safety I/O device types that do not have network capabilities. Examples include Safety sensors and actuators

### 6-6.1 Object Model

The Object Model in Figure 6-6.1 represents a Safety Discrete I/O Device. The Table below includes:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-6.1 Objects Present in a Safety Discrete I/O Device**

Object Class	Option/Required	# of Instances
Identity	Required	1
Message Router	Required	1
Link Specific Object (s)	Required	1
DeviceNet	Required for DeviceNet Safety	1
Connection Control	Required	#
Connection	Required for DeviceNet Safety	1
Safety Validator	Required	#
Safety Supervisor	Required	1
Assembly	Required	*
Discrete Input Point (DIP)	**	*
Safety Discrete Input Point (SDIP)	****	*
Discrete Output Point (DOP)	***	*
Safety Discrete Output Point (SDOP)	*****	*
Safety Dual Channel Output (SDCO)	##	*
Discrete Input Group (DIG)	Optional	1
Safety Discrete Input Group (SDIG)	Optional	1
Discrete Output Group (DOG)	Optional	1
Safety Discrete Output Group (SDOG)	Optional	1

\* = # of instances depends on the level of I/O support provided by the product.

\*\* = Optional for Standard Input Functions (provides backward compatibility)

\*\*\* = Optional for Standard Output Functions

\*\*\*\* = Required for Safety Input Functions

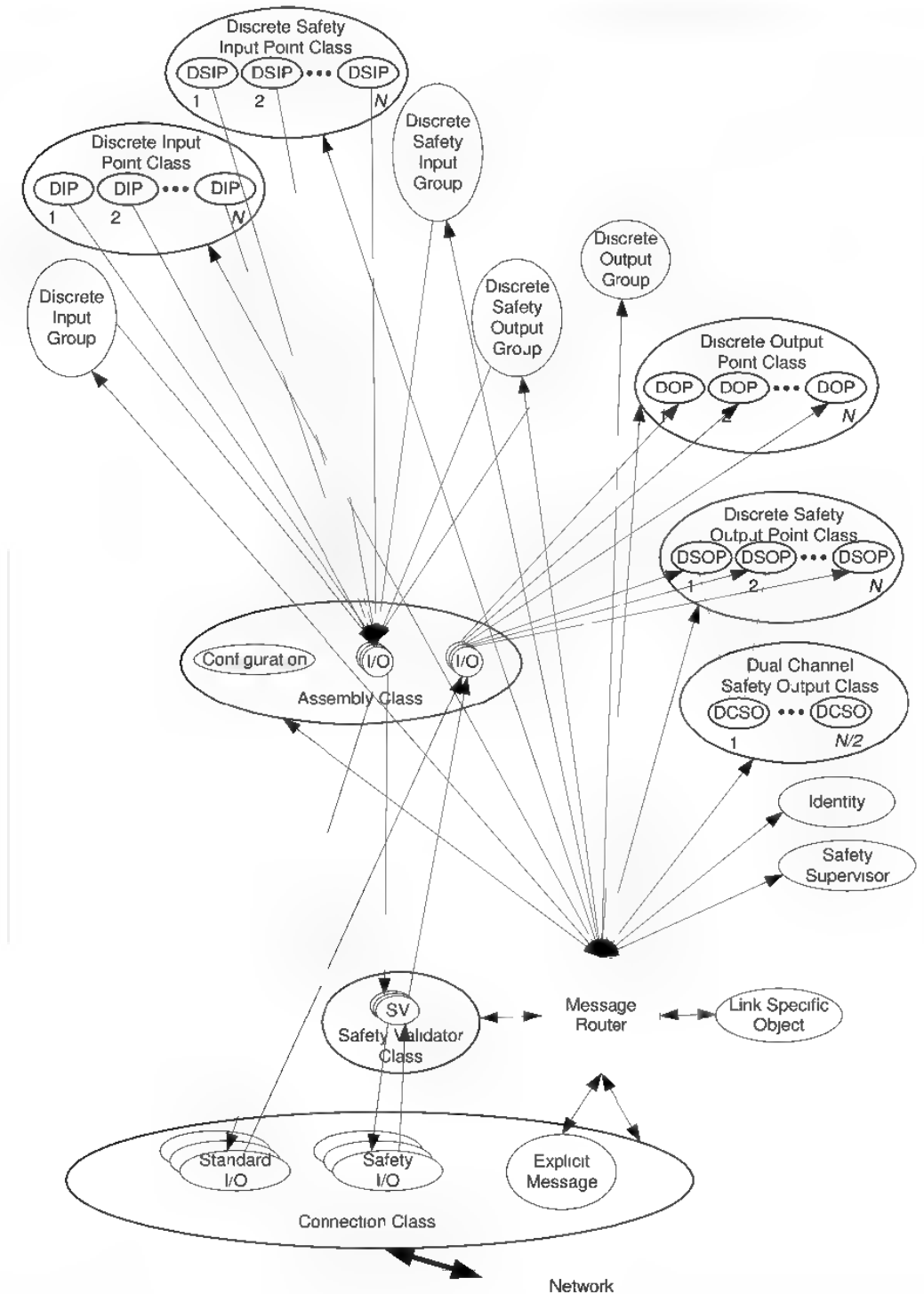
\*\*\*\*\* = Required for Safety Output Functions

# = Depends on the level of communications support provided by the product.

## = Required for Safety Output Dual Channel Functions



Figure 6-6.1 Object Model for Safety Discrete I/O Device





## 6-6.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-6.2 Object Affect on Behavior**

Object	Effect on Behavior
Identity	No effect
Message Router	No effect
Link Specific Object	Configures communication link attributes
Connection Control Object	Contains the logical ports into and out of the device
Assembly	Defined I/O data format and configuration data format
Safety Supervisor	Implements the Safety Network Configuration Tool Interface
Safety Validator	Handles safety protocol
Discrete Input Point (DIP)	Defines behavior of the discrete standard input points for this device
Safety Discrete Input Point (SDIP)	Defines behavior of the discrete safety input points for this device
Discrete Output Point (DOP)	Defines behavior of the discrete standard output points for this device
Safety Discrete Output Point (SDOP)	Defines behavior of the discrete safety output points for this device
Safety Dual Channel Output	Defines behavior of the dual channel discrete safety outputs for this device
Discrete Input Group	Stores the combined status of the Discrete Input Points
Safety Discrete Input Group	Stores the combined status of the Discrete Safety Input Points
Discrete Output Group	Stores the combined status of the Discrete Output Points
Safety Discrete Output Group	Stores the combined status of the Discrete Safety Output Points

### 6-6.2.1 Safety Supervisor Requirements

The safety supervisor definition contains a number of “profile dependent” functions. This section defines what the required behavior shall be for the Safety Discrete I/O device profile.

**Table 6-6.3 Safety Supervisor Implementation Level**

Implementation Level	Profile Requirement
Baseline functionality	Required
SNCT functionality	Optional



**Table 6-6.4 Safety Supervisor Profile-dependent State Event Behavior**

Event	“IDLE” State Behavior	“Executing” State Behavior	Comments
Safety Connection Failed/Closed	Remain in IDLE	If any standard or safety I/O connection still open, remain in EXECUTING, Else, Transition to IDLE	In this profile, device is in IDLE unless at least one standard or Safety I/O connection is established
Standard or Safety I/O Connection established	Transition to EXECUTING	Remain in EXECUTING state	
Type 1 SafetyOpen	Configure device, transition to EXECUTING	Drop standard and safety I/O connections, configure device, return to EXECUTING	
Safety I/O Connection Deleted	Not supported	Not supported	Connection deletion not supported
Mode Change	Error response: “Service Not Supported”	Error response: “Service Not Supported”	No mode change defined for this profile

### 6-6.3 Defining Object Interfaces

The objects in this device have the interfaces listed in the following table.

**Table 6-6.5 Object Interfaces**

Object	Interface
Identity	Message Router
Message Router	Explicit Messaging Connection Instance
DeviceNet	Message Router
Connection	Message Router
Assembly	I/O Connection (for standard connections) or Safety Validator (for Safety I/O connections) or Message Router
Safety Supervisor	Message Router
Safety Validator	Message Router or Safety I/O Connection
Discrete Input Point (DIP)	Message Router
Safety Discrete Input Point (SDIP)	Message Router or Assembly Object
Discrete Output Point (DOP)	Message Router or Assembly Object
Safety Discrete Output Point (SDOP)	Message Router or Assembly Object
Safety Dual Channel Output	Message Router or Assembly Object
Discrete Input Group	Message Router
Safety Discrete Input Group	Message Router
Discrete Output Group	Message Router
Safety Discrete Output Group	Message Router



#### **6-6.4 Relationship between DIP and SDIP**

A module using the Safety Discrete I/O profile may support either the SDIP, DIP, or both. An instance of the SDIP can be used to access inputs via safety evaluated data, or to access inputs as standard inputs without safety evaluation, or to access both. An Instance of the DIP can be used as standard discrete input data which is equivalent to the SDIP standard input data without safety evaluation.

#### **6-6.5 Relationship between DOP and SDOP**

A module using the Safety Discrete I/O profile may support either the SDOP, DOP, or both. The DOP may be used for standard Output functions or for Test Output functions.

#### **6-6.6 I/O Assembly Instances**

The I/O Assemblies for the Safety Discrete I/O Device may be classified into the following types.

**Table 6-6.6 I/O Assembly Object Instances for the Safety Discrete I/O Device**

<b>Input or Output</b>	<b>Data Type</b>
Input	Standard Data Only
Input	Safety Data Only
Input	Safety and Standard Data
Output	Standard Data Only
Output	Safety Data Only
Output	Safety and Standard Data

The assembly instance definition will differentiate safety data from standard data.

When an input assembly can be accessed by both a Safety I/O connection and/or a Standard I/O connection, the same instance number will be used by either I/O connection type.

When an output assembly can be accessed by a Safety I/O connection or a Standard I/O connection, the same instance number will be used by either of the I/O connection types. Only one I/O connection can be made to an output assembly at any given time.

Where ever possible the Standard Data only assemblies that are mapped to the DIP, and DOP objects will use the definition in the General Purpose Discrete I/O Profile (Device Type = 07<sub>hex</sub>). In addition this profile will define Standard Data only assemblies that are mapped to the SDIP that provide comparable functionality, but don't require the support of or the additional configuration of the DIP. These assemblies will be defined in the 180-1FF<sub>hex</sub> instance range.

**NOTE:** The Standard I/O assemblies defined in this device type have bits designated as "Reserved" within these I/O assemblies. These bits are defined as "reserved for internal use". The reserved bits within the I/O assemblies may be used for module specific internal purposes. An example of these would be diagnostics. The reserved bits should not be assumed to be zero on input assemblies.



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

As denoted in the table below instance number ranges 01-63<sub>hex</sub> and 100-17F<sub>hex</sub> of this profile will refer to the General Purpose Discrete I/O Profile. Those assembly definitions and mapping will not be repeated in this profile.

**Table 6-6.7 Instance Number Ranges**

Instance Range	Assembly Instance ID Range from Assembly Object Table 6.A	Qty	Data Type	Profile that defines the Assembly Instance	Typical Uses
01-63 <sub>hex</sub>	Open (static assemblies defined in device profile)	99	Standard Data Only	General Purpose Discrete I/O (07 <sub>hex</sub> )	DIP or DOP based I/O data
64-C7 <sub>hex</sub>	Vendor Specific static assemblies and dynamic assemblies	100	Vendor Specific	Vendor Specific	DIP or DOP based I/O data
C8-FF <sub>hex</sub>	Reserved by CIP for future use	56	-	-	
100-17F <sub>hex</sub>	Open (static assemblies defined in device profile)	128	Standard Data Only	General Purpose Discrete I/O (07 <sub>hex</sub> )	DIP or DOP based I/O data
180-1FF <sub>hex</sub>	Open (static assemblies defined in device profile)	128	Standard Data Only	Safety Discrete I/O (this profile)	SDIP or SDOP based standard I/O data
200-27F <sub>hex</sub>	Open (static assemblies defined in device profile)	128	Safety Data Only	Safety Discrete I/O (this profile)	SDIP or SDOP based safety I/O data
280-2FF <sub>hex</sub>	Open (static assemblies defined in device profile)	128	Safety and Standard Data	Safety Discrete I/O (this profile)	SDIP or SDOP based safety and standard I/O data
300-4FF <sub>hex</sub>	Vendor Specific static assemblies and dynamic assemblies	512	Vendor Specific	Vendor Specific	
500-FFFF <sub>hex</sub>	Reserved by CIP for future use	64,256	-	-	

As denoted in the table above instance number range 180-2FF<sub>hex</sub> will be defined in this profile.

The following Safety I/O assemblies 201-299<sub>hex</sub> are intended to line up with the Device Type 07<sub>hex</sub> assemblies 01-99<sub>dec</sub>, but the radix is different. In general the 2x0-2x9<sub>hex</sub> values will align with the x1-x9<sub>dec</sub> values. The 2xA-2xF<sub>hex</sub> assemblies are available for new definition. The Device Type 07, assemblies 7, 17, 27, 37, 47 and 57, that define *N* points are not supported by the 2x7<sub>hex</sub> Safety I/O assemblies. Since the *N* value is not contained in the safety forward open, these assembly definitions are ambiguous.

This device type uses the term Combined Status in place of the term Single Status used in Device Type 07<sub>hex</sub>. Combined Status is used because Single Status led to confusion whether it meant single status bit or status for a single point. This device type also uses the term Individual Status in place of the term Multiple Status used in Device Type 07<sub>hex</sub>. Individual Status is used because Multiple Status led to confusion whether it meant multiple status bits or status for multiple points.



**6-6.6.1 Standard Data Only Open Assemblies**

The following standard data only assemblies will be defined to access the Safety I/O points as standard I/O points without having to support the DIP object. The Standard Input(*n*) data in these assemblies maps to the *Standard Input Value* attribute of the SDIP, as opposed to the *Safety Input Logical Value* attribute being mapped to the assemblies including Safety Input(*n*) data. The Standard Input(*n*) data in these assemblies will be equivalent to the Discrete Input(*n*) data that is defined in the General Purpose Discrete I/O (07<sub>hex</sub>) device type. The Discrete Input(*n*) data maps to the *Value* attribute of the DIP.

**Table 6-6.8 Standard Data Only Open Assembly Instances**

Instance Number	Type	Standard Inputs
		Standard Input( <i>n</i> )
181 <sub>hex</sub>	In	1
182 <sub>hex</sub>	In	2
183 <sub>hex</sub>	In	4
184 <sub>hex</sub>	In	8
185 <sub>hex</sub>	In	16
186 <sub>hex</sub>	In	32
18A <sub>hex</sub>	In	6
18B <sub>hex</sub>	In	10
18C <sub>hex</sub>	In	12
18D <sub>hex</sub>	In	14



## 6-6.6.2 Safety Data Only Open Assemblies

Table 6-6.9 Safety Data Only Open Assembly Instances

Instance Number	Type	Safety Data		
		Safety I/O	Combined Status	Individual Status
		Safety Input (n)	Safety Input Status	Safety Input (n) Status
201 <sub>hex</sub>	In	1		
202 <sub>hex</sub>	In	2		
203 <sub>hex</sub>	In	4		
204 <sub>hex</sub>	In	8		
205 <sub>hex</sub>	In	16		
206 <sub>hex</sub>	In	32		
20A <sub>hex</sub>	In	6		
20B <sub>hex</sub>	In	10		
20C <sub>hex</sub>	In	12		
20D <sub>hex</sub>	In	14		
211 <sub>hex</sub>	In	1	Yes	
212 <sub>hex</sub>	In	2	Yes	
213 <sub>hex</sub>	In	4	Yes	
214 <sub>hex</sub>	In	8	Yes	
215 <sub>hex</sub>	In	16	Yes	
216 <sub>hex</sub>	In	32	Yes	
21A <sub>hex</sub>	In	6	Yes	
21B <sub>hex</sub>	In	10	Yes	
21C <sub>hex</sub>	In	12	Yes	
21D <sub>hex</sub>	In	14	Yes	
221 <sub>hex</sub>	In	1		1
222 <sub>hex</sub>	In	2		2
223 <sub>hex</sub>	In	4		4
224 <sub>hex</sub>	In	8		8
225 <sub>hex</sub>	In	16		16
226 <sub>hex</sub>	In	32		32
22A <sub>hex</sub>	In	6		6
22B <sub>hex</sub>	In	10		10
22C <sub>hex</sub>	In	12		12
22D <sub>hex</sub>	In	14		14



Safety Discrete I/O Device, Type: 23<sub>Hex</sub>

Table 6-6.10 Safety Data Only Open Assembly Instances

Instance Number	Type	Safety Data					
		Safety I/O		Combined Status		Individual Status	
		Safety Input(n)	Safety Output(n)	Safety Input Status	Safety Output Status	Safety Input(n) Status	Safety Output(n) Status
231 <sub>hex</sub>	Out	-	1				
232 <sub>hex</sub>	Out	-	2				
233 <sub>hex</sub>	Out	-	4				
234 <sub>hex</sub>	Out	-	8				
235 <sub>hex</sub>	Out	-	16				
236 <sub>hex</sub>	Out	-	32				
23A <sub>hex</sub>	Out	-	6				
23B <sub>hex</sub>	Out	-	10				
23C <sub>hex</sub>	Out	-	12				
23D <sub>hex</sub>	Out	-	14				
241 <sub>hex</sub>	In	-					1
242 <sub>hex</sub>	In	-					2
243 <sub>hex</sub>	In	-					4
244 <sub>hex</sub>	In	-					8
245 <sub>hex</sub>	In	-					16
246 <sub>hex</sub>	In	-					32
24A <sub>hex</sub>	In	-					6
24B <sub>hex</sub>	In	-					10
24C <sub>hex</sub>	In	-					12
24D <sub>hex</sub>	In	-					14
252 <sub>hex</sub>	In	2	*	Yes	Yes		
253 <sub>hex</sub>	In	4	*	Yes	Yes		
254 <sub>hex</sub>	In	8	*	Yes	Yes		
255 <sub>hex</sub>	In	16	*	Yes	Yes		
256 <sub>hex</sub>	In	32	*	Yes	Yes		
25A <sub>hex</sub>	In	6	*	Yes	Yes		
25B <sub>hex</sub>	In	10	*	Yes	Yes		
25C <sub>hex</sub>	In	12	*	Yes	Yes		
25D <sub>hex</sub>	In	14	*	Yes	Yes		
262 <sub>hex</sub>	In	2				2	2
263 <sub>hex</sub>	In	4				4	4
264 <sub>hex</sub>	In	8				8	8
265 <sub>hex</sub>	In	16				16	16
26A <sub>hex</sub>	In	6				6	6
26B <sub>hex</sub>	In	10				10	10
26C <sub>hex</sub>	In	12				12	12
26D <sub>hex</sub>	In	14				14	14



Table 6-6.11 Safety Data Only Open Assembly Instances

Instance Number	Type	Safety Data		
		Safety I/O	Combined Status	Individual Status
		Safety Input(n)	Safety Input Status	Safety Output(n) Status
270 <sub>hex</sub>	In	1	Yes	1
271 <sub>hex</sub>	In	2	Yes	1
272 <sub>hex</sub>	In	2	Yes	2
273 <sub>hex</sub>	In	4	Yes	2
274 <sub>hex</sub>	In	4	Yes	4
275 <sub>hex</sub>	In	8	Yes	4
276 <sub>hex</sub>	In	8	Yes	8
277 <sub>hex</sub>	In	16	Yes	8
278 <sub>hex</sub>	In	16	Yes	16
27A <sub>hex</sub>	In	6	Yes	2
27B <sub>hex</sub>	In	10	Yes	4
27C <sub>hex</sub>	In	12	Yes	4
27D <sub>hex</sub>	In	14	Yes	4

### 6-6.6.3 Safety Data and Standard Data Open Assemblies

Table 6-6.12 Safety Data & Standard Data Open Assembly Instances

Instance Number	Type	Safety Data	Standard Data	
		Safety I/O	Standard Inputs	Individual Safety Output Monitor
		Safety Input (n)	Standard Input(n)	Safety Output(n) Monitor
281 <sub>hex</sub>	In	1	1	
282 <sub>hex</sub>	In	2	2	
283 <sub>hex</sub>	In	4	4	
284 <sub>hex</sub>	In	8	8	
285 <sub>hex</sub>	In	16	16	
286 <sub>hex</sub>	In	32	32	
28A <sub>hex</sub>	In	6	6	
28B <sub>hex</sub>	In	10	10	
28C <sub>hex</sub>	In	12	12	
28D <sub>hex</sub>	In	14	14	
291 <sub>hex</sub>	In	1		1
292 <sub>hex</sub>	In	2		2
293 <sub>hex</sub>	In	4		4
294 <sub>hex</sub>	In	8		8
295 <sub>hex</sub>	In	16		16
296 <sub>hex</sub>	In	32		32
29A <sub>hex</sub>	In	6		6
29B <sub>hex</sub>	In	10		10
29C <sub>hex</sub>	In	12		12
29D <sub>hex</sub>	In	14		14



Table 6-6.13 Safety Data & Standard Data Only Open Assembly Instances

Instance Number	Type	Safety Data					Standard Data	
		Safety I/O		Combined Status	Individual Status		Individual Safety Output Monitor	Standard Outputs
		Safety Input(n)	Safety Output(n)	Safety Input Status	Safety Input(n) Status	Safety Output(n) Status	Safety Output(n) Monitor	Standard Output(n)
2A1 <sub>hex</sub>	In	1		Yes			1	
2A2 <sub>hex</sub>	In	2		Yes			2	
2A3 <sub>hex</sub>	In	4		Yes			4	
2A4 <sub>hex</sub>	In	8		Yes			8	
2A5 <sub>hex</sub>	In	16		Yes			16	
2A6 <sub>hex</sub>	In	32		Yes			32	
2AA <sub>hex</sub>	In	6		Yes			6	
2AB <sub>hex</sub>	In	10		Yes			10	
2AC <sub>hex</sub>	In	12		Yes			12	
2AD <sub>hex</sub>	In	14		Yes			14	
2B1 <sub>hex</sub>	In	1			1	1	1	
2B2 <sub>hex</sub>	In	2			2	2	2	
2B3 <sub>hex</sub>	In	4			4	4	4	
2B4 <sub>hex</sub>	In	8			8	8	8	
2B5 <sub>hex</sub>	In	16			16	16	16	
2B6 <sub>hex</sub>	In	32			32	32	32	
2BA <sub>hex</sub>	In	6			6	6	6	
2BB <sub>hex</sub>	In	10			10	10	10	
2BC <sub>hex</sub>	In	12			12	12	12	
2BD <sub>hex</sub>	In	14			14	14	14	
2C1 <sub>hex</sub>	Out		1					1
2C2 <sub>hex</sub>	Out		2					2
2C3 <sub>hex</sub>	Out		4					4
2C4 <sub>hex</sub>	Out		8					8
2C5 <sub>hex</sub>	Out		16					16
2C6 <sub>hex</sub>	Out		32					32
2CA <sub>hex</sub>	Out		6					6
2CB <sub>hex</sub>	Out		10					10
2CC <sub>hex</sub>	Out		12					12
2CD <sub>hex</sub>	Out		14					14



## 6-6.7 I/O Assembly Data Attribute Format

### 6-6.7.1 Standard Data Only Open Assemblies

The following standard data only assemblies will be defined to access the Safety I/O points as standard I/O points without having to support the DIP object. The Standard Input(*n*) data in these assemblies maps to the *Standard Input Value* attribute of the SDIP, as opposed to the *Safety Input Logical Value* attribute being mapped to the assemblies that include Safety Input(*n*) data. The Standard Input(*n*) data in these assemblies will be equivalent to the Discrete Input(*n*) data that is defined in the General Purpose Discrete I/O (07<sub>hex</sub>) device type. The Discrete Input(*n*) data maps to the *Value* attribute of the DIP.

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
181 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Standard Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
182 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Standard Input2	Standard Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
183 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Standard Input4	Standard Input3	Standard Input2	Standard Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
184 <sub>hex</sub>	0	Standard Input8	Standard Input7	Standard Input6	Standard Input5	Standard Input4	Standard Input3	Standard Input2	Standard Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
185 <sub>hex</sub>	0	Standard Input8	Standard Input7	Standard Input6	Standard Input5	Standard Input4	Standard Input3	Standard Input2	Standard Input1
	1	Standard Input16	Standard Input15	Standard Input14	Standard Input13	Standard Input12	Standard Input11	Standard Input10	Standard Input9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
186 <sub>hex</sub>	0	Standard Input8	Standard Input7	Standard Input6	Standard Input5	Standard Input4	Standard Input3	Standard Input2	Standard Input1
	1	Standard Input16	Standard Input15	Standard Input14	Standard Input13	Standard Input12	Standard Input11	Standard Input10	Standard Input9
	2	Standard Input24	Standard Input23	Standard Input22	Standard Input21	Standard Input20	Standard Input19	Standard Input18	Standard Input17
	3	Standard Input32	Standard Input31	Standard Input30	Standard Input29	Standard Input28	Standard Input27	Standard Input26	Standard Input25

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
18A <sub>hex</sub>	0	Reserved	Reserved	Standard Input6	Standard Input5	Standard Input4	Standard Input3	Standard Input2	Standard Input1



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
18B <sub>hex</sub>	0	Standard Input8	Standard Input7	Standard Input6	Standard Input5	Standard Input4	Standard Input3	Standard Input2	Standard Input1
	1	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Standard Input10	Standard Input9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
18C <sub>hex</sub>	0	Standard Input8	Standard Input7	Standard Input6	Standard Input5	Standard Input4	Standard Input3	Standard Input2	Standard Input1
	1	Reserved	Reserved	Reserved	Reserved	Standard Input12	Standard Input11	Standard Input10	Standard Input9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
18D <sub>hex</sub>	0	Standard Input8	Standard Input7	Standard Input6	Standard Input5	Standard Input4	Standard Input3	Standard Input2	Standard Input1
	1	Reserved	Reserved	Standard Input14	Standard Input13	Standard Input12	Standard Input11	Standard Input10	Standard Input9

### 6-6.7.2 Safety Data Only Open Assemblies

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
201 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
202 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
203 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Safety Input4	Safety Input3	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
204 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
205 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
206 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input24	Safety Input23	Safety Input22	Safety Input21	Safety Input20	Safety Input19	Safety Input18	Safety Input17
	3	Safety Input32	Safety Input31	Safety Input30	Safety Input29	Safety Input28	Safety Input27	Safety Input26	Safety Input25



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
20A <sub>hex</sub>	0	Reserved	Reserved	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
20B <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Input10	Safety Input9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
20C <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Reserved	Reserved	Reserved	Reserved	Safety Input12	Safety Input11	Safety Input10	Safety Input9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
20D <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Reserved	Reserved	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
211 <sub>hex</sub>	0	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
212 <sub>hex</sub>	0	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
213 <sub>hex</sub>	0	Safety Input Status	Reserved	Reserved	Reserved	Safety Input4	Safety Input3	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
214 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
215 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
216 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input24	Safety Input23	Safety Input22	Safety Input21	Safety Input20	Safety Input19	Safety Input18	Safety Input17
	3	Safety Input32	Safety Input31	Safety Input30	Safety Input29	Safety Input28	Safety Input27	Safety Input26	Safety Input25
	4	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
21A <sub>hex</sub>	0	Safety Input Status	Reserved	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
21B <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Input10	Safety Input9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
21C <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Reserved	Reserved	Safety Input12	Safety Input11	Safety Input10	Safety Input9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
21D <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
221 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Input1 Status	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
222 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Safety Input2 Status	Safety Input1 Status	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
223 <sub>hex</sub>	0	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status	Safety Input4	Safety Input3	Safety Input2	Safety Input1



## Volume 5: CIP Safety, Chapter 6: Device Profiles

### Safety Discrete I/O Device, Type: 23<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
224 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
225 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status
	3	Safety Input16 Status	Safety Input15 Status	Safety Input14 Status	Safety Input13 Status	Safety Input12 Status	Safety Input11 Status	Safety Input10 Status	Safety Input9 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
226 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input24	Safety Input23	Safety Input22	Safety Input21	Safety Input20	Safety Input19	Safety Input18	Safety Input17
	3	Safety Input32	Safety Input31	Safety Input30	Safety Input29	Safety Input28	Safety Input27	Safety Input26	Safety Input25
	4	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status
	5	Safety Input16 Status	Safety Input15 Status	Safety Input14 Status	Safety Input13 Status	Safety Input12 Status	Safety Input11 Status	Safety Input10 Status	Safety Input9 Status
	6	Safety Input24 Status	Safety Input23 Status	Safety Input22 Status	Safety Input21 Status	Safety Input20 Status	Safety Input19 Status	Safety Input18 Status	Safety Input17 Status
	7	Safety Input32 Status	Safety Input31 Status	Safety Input30 Status	Safety Input29 Status	Safety Input28 Status	Safety Input27 Status	Safety Input26 Status	Safety Input25 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
22A <sub>hex</sub>	0	Safety Input2 Status	Safety Input1 Status	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Reserved	Reserved	Reserved	Reserved	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
22B <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status	Safety Input10	Safety Input9
	2	Reserved	Reserved	Reserved	Reserved	Safety Input10 Status	Safety Input9 Status	Safety Input8 Status	Safety Input7 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
22C <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input12 Status	Safety Input11 Status	Safety Input10 Status	Safety Input9 Status	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
22D <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input2 Status	Safety Input1 Status	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input10 Status	Safety Input9 Status	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status
	3	Reserved	Reserved	Reserved	Reserved	Safety Input14 Status	Safety Input13 Status	Safety Input12 Status	Safety Input11 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
231 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
232 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output2	Safety Output1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
233 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Safety Output4	Safety Output3	Safety Output2	Safety Output1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
234 <sub>hex</sub>	0	Safety Output8	Safety Output7	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
235 <sub>hex</sub>	0	Safety Output8	Safety Output7	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1
	1	Safety Output16	Safety Output15	Safety Output14	Safety Output13	Safety Output12	Safety Output11	Safety Output10	Safety Output9



# Volume 5: CIP Safety, Chapter 6: Device Profiles

## Safety Discrete I/O Device, Type: 23<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
236 <sub>hex</sub>	0	Safety Output8	Safety Output7	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1
	1	Safety Output16	Safety Output15	Safety Output14	Safety Output13	Safety Output12	Safety Output11	Safety Output10	Safety Output9
	2	Safety Output24	Safety Output23	Safety Output22	Safety Output21	Safety Output20	Safety Output19	Safety Output18	Safety Output17
	3	Safety Output32	Safety Output31	Safety Output30	Safety Output29	Safety Output28	Safety Output27	Safety Output26	Safety Output25

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
23A <sub>hex</sub>	0	Reserved	Reserved	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
23B <sub>hex</sub>	0	Safety Output8	Safety Output7	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1
	1	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output10	Safety Output9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
23C <sub>hex</sub>	0	Safety Output8	Safety Output7	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1
	1	Reserved	Reserved	Reserved	Reserved	Safety Output12	Safety Output11	Safety Output10	Safety Output9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
23D <sub>hex</sub>	0	Safety Output8	Safety Output7	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1
	1	Reserved	Reserved	Safety Output14	Safety Output13	Safety Output12	Safety Output11	Safety Output10	Safety Output9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
241 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
242 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output2 Status	Safety Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
243 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
244 <sub>hex</sub>	0	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status



# Volume 5: CIP Safety, Chapter 6: Device Profiles

## Safety Discrete I/O Device, Type: 23<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
245 <sub>hex</sub>	0	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
	1	Safety Output16 Status	Safety Output15 Status	Safety Output14 Status	Safety Output13 Status	Safety Output12 Status	Safety Output11 Status	Safety Output10 Status	Safety Output9 Status
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
246 <sub>hex</sub>	0	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
	1	Safety Output16 Status	Safety Output15 Status	Safety Output14 Status	Safety Output13 Status	Safety Output12 Status	Safety Output11 Status	Safety Output10 Status	Safety Output9 Status
	2	Safety Output24 Status	Safety Output23 Status	Safety Output22 Status	Safety Output21 Status	Safety Output20 Status	Safety Output19 Status	Safety Output18 Status	Safety Output17 Status
	3	Safety Output32 Status	Safety Output31 Status	Safety Output30 Status	Safety Output29 Status	Safety Output28 Status	Safety Output27 Status	Safety Output26 Status	Safety Output25 Status
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
24A <sub>hex</sub>	0	Reserved	Reserved	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
24B <sub>hex</sub>	0	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
	1	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output10 Status	Safety Output9 Status
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
24C <sub>hex</sub>	0	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
	1	Reserved	Reserved	Reserved	Reserved	Safety Output12 Status	Safety Output11 Status	Safety Output10 Status	Safety Output9 Status
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
24D <sub>hex</sub>	0	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
	1	Reserved	Reserved	Safety Output14 Status	Safety Output13 Status	Safety Output12 Status	Safety Output11 Status	Safety Output10 Status	Safety Output9 Status
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
252 <sub>hex</sub>	0	Safety Input Status	Safety Output Status	Reserved	Reserved	Reserved	Reserved	Safety Input2	Safety Input1



# Volume 5: CIP Safety, Chapter 6: Device Profiles

## Safety Discrete I/O Device, Type: 23<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
253 <sub>hex</sub>	0	Safety Input Status	Safety Output Status	Reserved	Reserved	Safety Input4	Safety Input3	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
254 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Safety Output Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
255 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input Status	Safety Output Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
256 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input24	Safety Input23	Safety Input22	Safety Input21	Safety Input20	Safety Input19	Safety Input18	Safety Input17
	3	Safety Input32	Safety Input31	Safety Input30	Safety Input29	Safety Input28	Safety Input27	Safety Input26	Safety Input25
	4	Safety Input Status	Safety Output Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
25A <sub>hex</sub>	0	Safety Input Status	Safety Output Status	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
25B <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Safety Output Status	Reserved	Reserved	Reserved	Reserved	Safety Input10	Safety Input9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
25C <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Safety Output Status	Reserved	Reserved	Safety Input12	Safety Input11	Safety Input10	Safety Input9



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
25D <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Safety Output Status	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
262 <sub>hex</sub>	0	Reserved	Reserved	Safety Output2 Status	Safety Output1 Status	Safety Input2 Status	Safety Input1 Status	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
263 <sub>hex</sub>	0	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Reserved	Reserved	Reserved	Reserved	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
264 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status
	2	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
265 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status
	3	Safety Input16 Status	Safety Input15 Status	Safety Input14 Status	Safety Input13 Status	Safety Input12 Status	Safety Input11 Status	Safety Input10 Status	Safety Input9 Status
	4	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
	5	Safety Output16 Status	Safety Output15 Status	Safety Output14 Status	Safety Output13 Status	Safety Output12 Status	Safety Output11 Status	Safety Output10 Status	Safety Output9 Status



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
26A <sub>hex</sub>	0	Safety Input2 Status	Safety Input1 Status	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status
	2	Reserved	Reserved	Reserved	Reserved	Safety Output4 Status	Safety Output3 Status	Safety Output6 Status	Safety Output5 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
26B <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status	Safety Input10	Safety Input9
	2	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status	Safety Input10 Status	Safety Input9 Status	Safety Input8 Status	Safety Input7 Status
	3	Reserved	Reserved	Safety Output10 Status	Safety Output9 Status	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
26C <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input12 Status	Safety Input11 Status	Safety Input10 Status	Safety Input9 Status	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status
	3	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
	4	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output10 Status	Safety Output9 Status



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
26D <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input2 Status	Safety Input1 Status	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input10 Status	Safety Input9 Status	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status
	3	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status	Safety Input14 Status	Safety Input13 Status	Safety Input12 Status	Safety Input11 Status
	4	Safety Output12 Status	Safety Output11 Status	Safety Output10 Status	Safety Output9 Status	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status
	5	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output14 Status	Safety Output13 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
270 <sub>hex</sub>	0	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output1 Status	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
271 <sub>hex</sub>	0	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Safety Output1 Status	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
272 <sub>hex</sub>	0	Safety Input Status	Reserved	Reserved	Reserved	Safety Output2 Status	Safety Output1 Status	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
273 <sub>hex</sub>	0	Safety Input Status	Reserved	Safety Output2 Status	Safety Output1 Status	Safety Input4	Safety Input3	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
274 <sub>hex</sub>	0	Safety Input Status	Reserved	Reserved	Reserved	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Reserved	Reserved	Reserved	Reserved	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
275 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Reserved	Reserved	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
276 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	2	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
277 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	3	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
278 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	3	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
	4	Safety Output16 Status	Safety Output15 Status	Safety Output14 Status	Safety Output13 Status	Safety Output12 Status	Safety Output11 Status	Safety Output10 Status	Safety Output9 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
27A <sub>hex</sub>	0	Safety Input Status	Reserved	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output2 Status	Safety Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
27B <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status	Safety Input10	Safety Input9



Safety Discrete I/O Device, Type: 23<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
27C <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Reserved	Reserved	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Reserved	Reserved	Reserved	Reserved	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
27D <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	4	Reserved	Reserved	Reserved	Reserved	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status

### 6-6.7.3 Safety Data and Standard Data Open Assemblies

Note: Safety Data is bolded

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
281 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Standard Input1	<b>Safety Input1</b>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
282 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Standard Input2	Standard Input1	<b>Safety Input2</b>	<b>Safety Input1</b>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
283 <sub>hex</sub>	0	Standard Input4	Standard Input3	Standard Input2	Standard Input1	<b>Safety Input4</b>	<b>Safety Input3</b>	<b>Safety Input2</b>	<b>Safety Input1</b>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
284 <sub>hex</sub>	0	<b>Safety Input8</b>	<b>Safety Input7</b>	<b>Safety Input6</b>	<b>Safety Input5</b>	<b>Safety Input4</b>	<b>Safety Input3</b>	<b>Safety Input2</b>	<b>Safety Input1</b>
	1	Standard Input8	Standard Input7	Standard Input6	Standard Input5	Standard Input4	Standard Input3	Standard Input2	Standard Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
285 <sub>hex</sub>	0	<b>Safety Input8</b>	<b>Safety Input7</b>	<b>Safety Input6</b>	<b>Safety Input5</b>	<b>Safety Input4</b>	<b>Safety Input3</b>	<b>Safety Input2</b>	<b>Safety Input1</b>
	1	<b>Safety Input16</b>	<b>Safety Input15</b>	<b>Safety Input14</b>	<b>Safety Input13</b>	<b>Safety Input12</b>	<b>Safety Input11</b>	<b>Safety Input10</b>	<b>Safety Input9</b>
	2	Standard Input8	Standard Input7	Standard Input6	Standard Input5	Standard Input4	Standard Input3	Standard Input2	Standard Input1
	3	Standard Input16	Standard Input15	Standard Input14	Standard Input13	Standard Input12	Standard Input11	Standard Input10	Standard Input9



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
286 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input24	Safety Input23	Safety Input22	Safety Input21	Safety Input20	Safety Input19	Safety Input18	Safety Input17
	3	Safety Input32	Safety Input31	Safety Input30	Safety Input29	Safety Input28	Safety Input27	Safety Input26	Safety Input25
	4	Standard Input8	Standard Input7	Standard Input6	Standard Input5	Standard Input4	Standard Input3	Standard Input2	Standard Input1
	5	Standard Input16	Standard Input15	Standard Input14	Standard Input13	Standard Input12	Standard Input11	Standard Input10	Standard Input9
	6	Standard Input24	Standard Input23	Standard Input22	Standard Input21	Standard Input20	Standard Input19	Standard Input18	Standard Input17
	7	Standard Input32	Standard Input31	Standard Input30	Standard Input29	Standard Input28	Standard Input27	Standard Input26	Standard Input25

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
28A <sub>hex</sub>	0	Standard Input2	Standard Input1	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Reserved	Reserved	Reserved	Reserved	Standard Input6	Standard Input5	Standard Input4	Standard Input3

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
28B <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Standard Input6	Standard Input5	Standard Input4	Standard Input3	Standard Input2	Standard Input1	Safety Input10	Safety Input9
	2	Reserved	Reserved	Reserved	Reserved	Standard Input10	Standard Input9	Standard Input8	Standard Input7

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
28C <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Standard Input4	Standard Input3	Standard Input2	Standard Input1	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Standard Input12	Standard Input11	Standard Input10	Standard Input9	Standard Input8	Standard Input7	Standard Input6	Standard Input5

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
28D <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Standard Input2	Standard Input1	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Standard Input10	Standard Input9	Standard Input8	Standard Input7	Standard Input6	Standard Input5	Standard Input4	Standard Input3
	3	Reserved	Reserved	Reserved	Reserved	Standard Input14	Standard Input13	Standard Input12	Standard Input11

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
291 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output1 Monitor	Safety Input1



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
292 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Safety Output2 Monitor	Safety Output1 Monitor	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
293 <sub>hex</sub>	0	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor	Safety Input4	Safety Input3	Safety Input2	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
294 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
295 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor
	3	Safety Output16 Monitor	Safety Output15 Monitor	Safety Output14 Monitor	Safety Output13 Monitor	Safety Output12 Monitor	Safety Output11 Monitor	Safety Output10 Monitor	Safety Output9 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
296 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input24	Safety Input23	Safety Input22	Safety Input21	Safety Input20	Safety Input19	Safety Input18	Safety Input17
	3	Safety Input32	Safety Input31	Safety Input30	Safety Input29	Safety Input28	Safety Input27	Safety Input26	Safety Input25
	4	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor
	5	Safety Output16 Monitor	Safety Output15 Monitor	Safety Output14 Monitor	Safety Output13 Monitor	Safety Output12 Monitor	Safety Output11 Monitor	Safety Output10 Monitor	Safety Output9 Monitor
	6	Safety Output24 Monitor	Safety Output23 Monitor	Safety Output22 Monitor	Safety Output21 Monitor	Safety Output20 Monitor	Safety Output19 Monitor	Safety Output18 Monitor	Safety Output17 Monitor
	7	Safety Output32 Monitor	Safety Output31 Monitor	Safety Output30 Monitor	Safety Output29 Monitor	Safety Output28 Monitor	Safety Output27 Monitor	Safety Output26 Monitor	Safety Output25 Monitor



# Volume 5: CIP Safety, Chapter 6: Device Profiles

## Safety Discrete I/O Device, Type: 23<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
29A <sub>hex</sub>	0	Safety Output2 Monitor	Safety Output1 Monitor	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Reserved	Reserved	Reserved	Reserved	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
29B <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor	Safety Input10	Safety Input9
	2	Reserved	Reserved	Reserved	Reserved	Safety Output10 Monitor	Safety Output9 Monitor	Safety Output8 Monitor	Safety Output7 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
29C <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Output12 Monitor	Safety Output11 Monitor	Safety Output10 Monitor	Safety Output9 Monitor	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
29D <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Output2 Monitor	Safety Output1 Monitor	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Output10 Monitor	Safety Output9 Monitor	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor
	3	Reserved	Reserved	Reserved	Reserved	Safety Output14 Monitor	Safety Output13 Monitor	Safety Output12 Monitor	Safety Output11 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2A1 <sub>hex</sub>	0	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Input1
	1	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output1 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2A2 <sub>hex</sub>	0	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Input2	Safety Input1
	1	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output2 Monitor	Safety Output1 Monitor



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2A3 <sub>hex</sub>	0	Safety Input Status	Reserved	Reserved	Reserved	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Reserved	Reserved	Reserved	Reserved	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2A4 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	2	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2A5 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	3	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor
	4	Safety Output16 Monitor	Safety Output15 Monitor	Safety Output14 Monitor	Safety Output13 Monitor	Safety Output12 Monitor	Safety Output11 Monitor	Safety Output10 Monitor	Safety Output9 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2A6 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input24	Safety Input23	Safety Input22	Safety Input21	Safety Input20	Safety Input19	Safety Input18	Safety Input17
	3	Safety Input32	Safety Input31	Safety Input30	Safety Input29	Safety Input28	Safety Input27	Safety Input26	Safety Input25
	4	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	5	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor
	6	Safety Output16 Monitor	Safety Output15 Monitor	Safety Output14 Monitor	Safety Output13 Monitor	Safety Output12 Monitor	Safety Output11 Monitor	Safety Output10 Monitor	Safety Output9 Monitor
	7	Safety Output24 Monitor	Safety Output23 Monitor	Safety Output22 Monitor	Safety Output21 Monitor	Safety Output20 Monitor	Safety Output19 Monitor	Safety Output18 Monitor	Safety Output17 Monitor
	8	Safety Output32 Monitor	Safety Output31 Monitor	Safety Output30 Monitor	Safety Output29 Monitor	Safety Output28 Monitor	Safety Output27 Monitor	Safety Output26 Monitor	Safety Output25 Monitor



# Volume 5: CIP Safety, Chapter 6: Device Profiles

## Safety Discrete I/O Device, Type: 23<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2AA <sub>hex</sub>	0	Safety Input Status	Reserved	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Reserved	Reserved	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2AB <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Input10	Safety Input9
	2	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor
	3	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output10 Monitor	Safety Output9 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2AC <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Reserved	Reserved	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor
	3	Reserved	Reserved	Reserved	Reserved	Safety Output12 Monitor	Safety Output11 Monitor	Safety Output10 Monitor	Safety Output9 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2AD <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input Status	Reserved	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor
	3	Reserved	Reserved	Safety Output14 Monitor	Safety Output13 Monitor	Safety Output12 Monitor	Safety Output11 Monitor	Safety Output10 Monitor	Safety Output9 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2B1 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Safety Output1 Monitor	Safety Output1 Status	Safety Input1 Status	Safety Input1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2B2 <sub>hex</sub>	0	Safety Output2 Monitor	Safety Output1 Monitor	Safety Output2 Status	Safety Output1 Status	Safety Input2 Status	Safety Input1 Status	Safety Input2	Safety Input1



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2B3 <sub>hex</sub>	0	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2B4 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status
	2	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
	3	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2B5 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status
	3	Safety Input16 Status	Safety Input15 Status	Safety Input14 Status	Safety Input13 Status	Safety Input12 Status	Safety Input11 Status	Safety Input10 Status	Safety Input9 Status
	4	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
	5	Safety Output16 Status	Safety Output15 Status	Safety Output14 Status	Safety Output13 Status	Safety Output12 Status	Safety Output11 Status	Safety Output10 Status	Safety Output9 Status
	6	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor
	7	Safety Output16 Monitor	Safety Output15 Monitor	Safety Output14 Monitor	Safety Output13 Monitor	Safety Output12 Monitor	Safety Output11 Monitor	Safety Output10 Monitor	Safety Output9 Monitor



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2B6 <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input16	Safety Input15	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input24	Safety Input23	Safety Input22	Safety Input21	Safety Input20	Safety Input19	Safety Input18	Safety Input17
	3	Safety Input32	Safety Input31	Safety Input30	Safety Input29	Safety Input28	Safety Input27	Safety Input26	Safety Input25
	4	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status
	5	Safety Input16 Status	Safety Input15 Status	Safety Input14 Status	Safety Input13 Status	Safety Input12 Status	Safety Input11 Status	Safety Input10 Status	Safety Input9 Status
	6	Safety Input24 Status	Safety Input23 Status	Safety Input22 Status	Safety Input21 Status	Safety Input20 Status	Safety Input19 Status	Safety Input18 Status	Safety Input17 Status
	7	Safety Input32 Status	Safety Input31 Status	Safety Input30 Status	Safety Input29 Status	Safety Input28 Status	Safety Input27 Status	Safety Input26 Status	Safety Input25 Status
	8	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
	9	Safety Output16 Status	Safety Output15 Status	Safety Output14 Status	Safety Output13 Status	Safety Output12 Status	Safety Output11 Status	Safety Output10 Status	Safety Output9 Status
	10	Safety Output24 Status	Safety Output23 Status	Safety Output22 Status	Safety Output21 Status	Safety Output20 Status	Safety Output19 Status	Safety Output18 Status	Safety Output17 Status
	11	Safety Output32 Status	Safety Output31 Status	Safety Output30 Status	Safety Output29 Status	Safety Output28 Status	Safety Output27 Status	Safety Output26 Status	Safety Output25 Status
	12	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor
	13	Safety Output16 Monitor	Safety Output15 Monitor	Safety Output14 Monitor	Safety Output13 Monitor	Safety Output12 Monitor	Safety Output11 Monitor	Safety Output10 Monitor	Safety Output9 Monitor
	14	Safety Output24 Monitor	Safety Output23 Monitor	Safety Output22 Monitor	Safety Output21 Monitor	Safety Output20 Monitor	Safety Output19 Monitor	Safety Output18 Monitor	Safety Output17 Monitor
	15	Safety Output32 Monitor	Safety Output31 Monitor	Safety Output30 Monitor	Safety Output29 Monitor	Safety Output28 Monitor	Safety Output27 Monitor	Safety Output26 Monitor	Safety Output25 Monitor



# Volume 5: CIP Safety, Chapter 6: Device Profiles

## Safety Discrete I/O Device, Type: 23<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2BA <sub>hex</sub>	0	Safety Input2 Status	Safety Input1 Status	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status
	2	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output6 Status	Safety Output5 Status
	3	Reserved	Reserved	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2BB <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status	Safety Input10	Safety Input9
	2	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status	Safety Input10 Status	Safety Input9 Status	Safety Input8 Status	Safety Input7 Status
	3	Reserved	Reserved	Safety Output10 Status	Safety Output9 Status	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status
	4	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor
	5	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output10 Monitor	Safety Output9 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2BC <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input4 Status	Safety Input3 Status	Safety Input2 Status	Safety Input1 Status	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input12 Status	Safety Input11 Status	Safety Input10 Status	Safety Input9 Status	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status
	3	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status
	4	Reserved	Reserved	Reserved	Reserved	Safety Output12 Status	Safety Output11 Status	Safety Output10 Status	Safety Output9 Status
	5	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor
	6	Reserved	Reserved	Reserved	Reserved	Safety Output12 Monitor	Safety Output11 Monitor	Safety Output10 Monitor	Safety Output9 Monitor



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2BD <sub>hex</sub>	0	Safety Input8	Safety Input7	Safety Input6	Safety Input5	Safety Input4	Safety Input3	Safety Input2	Safety Input1
	1	Safety Input2 Status	Safety Input1 Status	Safety Input14	Safety Input13	Safety Input12	Safety Input11	Safety Input10	Safety Input9
	2	Safety Input10 Status	Safety Input9 Status	Safety Input8 Status	Safety Input7 Status	Safety Input6 Status	Safety Input5 Status	Safety Input4 Status	Safety Input3 Status
	3	Safety Output4 Status	Safety Output3 Status	Safety Output2 Status	Safety Output1 Status	Safety Input14 Status	Safety Input13 Status	Safety Input12 Status	Safety Input11 Status
	4	Safety Output12 Status	Safety Output11 Status	Safety Output10 Status	Safety Output9 Status	Safety Output8 Status	Safety Output7 Status	Safety Output6 Status	Safety Output5 Status
	5	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Safety Output14 Status	Safety Output13 Status
	6	Safety Output8 Monitor	Safety Output7 Monitor	Safety Output6 Monitor	Safety Output5 Monitor	Safety Output4 Monitor	Safety Output3 Monitor	Safety Output2 Monitor	Safety Output1 Monitor
	7	Reserved	Reserved	Safety Output14 Monitor	Safety Output13 Monitor	Safety Output12 Monitor	Safety Output11 Monitor	Safety Output10 Monitor	Safety Output9 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C1 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Standard Output1	Safety Output1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C2 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Standard Output2	Standard Output1	Safety Output2	Safety Output1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C3 <sub>hex</sub>	0	Standard Output4	Standard Output3	Standard Output2	Standard Output1	Safety Output4	Safety Output3	Safety Output2	Safety Output1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C4 <sub>hex</sub>	0	Safety Output8	Safety Output7	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1
	1	Standard Output8	Standard Output7	Standard Output6	Standard Output5	Standard Output4	Standard Output3	Standard Output2	Standard Output1

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C5 <sub>hex</sub>	0	Safety Output8	Safety Output7	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1
	1	Safety Output16	Safety Output15	Safety Output14	Safety Output13	Safety Output12	Safety Output11	Safety Output10	Safety Output9
	2	Standard Output8	Standard Output7	Standard Output6	Standard Output5	Standard Output4	Standard Output3	Standard Output2	Standard Output1
	3	Standard Output16	Standard Output15	Standard Output14	Standard Output13	Standard Output12	Standard Output11	Standard Output10	Standard Output9



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C6 <sub>hex</sub>	0	Safety Output8	Safety Output7	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1
	1	Safety Output16	Safety Output15	Safety Output14	Safety Output13	Safety Output12	Safety Output11	Safety Output10	Safety Output9
	2	Safety Output24	Safety Output23	Safety Output22	Safety Output21	Safety Output20	Safety Output19	Safety Output18	Safety Output17
	3	Safety Output32	Safety Output31	Safety Output30	Safety Output29	Safety Output28	Safety Output27	Safety Output26	Safety Output25
	4	Standard Output8	Standard Output7	Standard Output6	Standard Output5	Standard Output4	Standard Output3	Standard Output2	Standard Output1
	5	Standard Output16	Standard Output15	Standard Output14	Standard Output13	Standard Output12	Standard Output11	Standard Output10	Standard Output9
	6	Standard Output24	Standard Output23	Standard Output22	Standard Output21	Standard Output20	Standard Output19	Standard Output18	Standard Output17
	7	Standard Output32	Standard Output31	Standard Output30	Standard Output29	Standard Output28	Standard Output27	Standard Output26	Standard Output25

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2CA <sub>hex</sub>	0	Standard Output2	Standard Output1	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1
	1	Reserved	Reserved	Reserved	Reserved	Standard Output6	Standard Output5	Standard Output4	Standard Output3

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2CB <sub>hex</sub>	0	Safety Output8	Safety Output7	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1
	1	Standard Output6	Standard Output5	Standard Output4	Standard Output3	Standard Output2	Standard Output1	Safety Output10	Safety Output9
	2	Reserved	Reserved	Reserved	Reserved	Standard Output10	Standard Output9	Standard Output8	Standard Output7

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2CC <sub>hex</sub>	0	Safety Output8	Safety Output7	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1
	1	Standard Output4	Standard Output3	Standard Output2	Standard Output1	Safety Output12	Safety Output11	Safety Output10	Safety Output9
	2	Standard Output12	Standard Output11	Standard Output10	Standard Output9	Standard Output8	Standard Output7	Standard Output6	Standard Output5

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2CD <sub>hex</sub>	0	Safety Output8	Safety Output7	Safety Output6	Safety Output5	Safety Output4	Safety Output3	Safety Output2	Safety Output1
	1	Standard Output2	Standard Output1	Safety Output14	Safety Output13	Safety Output12	Safety Output11	Safety Output10	Safety Output9
	2	Standard Output10	Standard Output9	Standard Output8	Standard Output7	Standard Output6	Standard Output5	Standard Output4	Standard Output3
	3	Reserved	Reserved	Reserved	Reserved	Standard Output14	Standard Output13	Standard Output12	Standard Output11



## **6-6.8 Mapping I/O Assembly Data Attribute Components**

### **6-6.8.1 Standard Data Only Open Assemblies**

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the input assemblies with standard input data and no status.

**Table 6-6.14 Instance Numbers 181 -18D<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Standard Input/V	Safety Discrete Input Point	3D <sub>hex</sub>	N	Standard Input Value	10

### **6-6.8.2 Safety Data Only Open Assemblies**

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the input assemblies with safety input data and no safety status.

**Table 6-6.15 Instance Numbers 201 -20D<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Input/V	Safety Discrete Input Point	3D <sub>hex</sub>	N	Safety Input Logical Value	7

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the input assemblies with safety input data and combined safety input status.

**Table 6-6.16 Instance Numbers 211 -21D<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Input/V	Safety Discrete Input Point	3D <sub>hex</sub>	N	Safety Input Logical Value	7
Safety Input Status	Safety Discrete Input Group	3E <sub>hex</sub>	1	Status	5

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the input assemblies with safety input data and individual safety input status.

**Table 6-6.17 Instance Numbers 221 -22D<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Input/V	Safety Discrete Input Point	3D <sub>hex</sub>	N	Safety Input Logical Value	7
Safety Input/V Status	Safety Discrete Input Point	3D <sub>hex</sub>	N	Status	5



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the output assemblies with safety output data.

**Table 6-6.18 Instance Numbers 231 -23D<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Output <i>N</i>	Safety Discrete Output Point	3B <sub>hex</sub>	<i>N</i>	Safety Output Value	3

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the input assemblies with individual safety output status.

**Table 6-6.19 Instance Numbers 241 -24D<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Output/ <i>N</i> Status	Safety Discrete Output Point	3B <sub>hex</sub>	<i>N</i>	Status	5

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the input assemblies with safety input data, combined safety input status, and combined safety output status.

**Table 6-6.20 Instance Numbers 252 -25D<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Input/ <i>N</i>	Safety Discrete Input Point	3D <sub>hex</sub>	<i>N</i>	Safety Input Logical Value	7
Safety Input Status	Safety Discrete Input Group	3E <sub>hex</sub>	1	Status	5
Safety Output Status	Safety Discrete Output Group	3C <sub>hex</sub>	1	Status	5

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the input assemblies with safety input data, individual safety input status, and individual safety output status.

**Table 6-6.21 Instance Numbers 262 -26D<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Input/ <i>N</i>	Safety Discrete Input Point	3D <sub>hex</sub>	<i>N</i>	Safety Input Logical Value	7
Safety Input/ <i>N</i> Status	Safety Discrete Input Point	3D <sub>hex</sub>	<i>N</i>	Status	5
Safety Output/ <i>N</i> Status	Safety Discrete Output Point	3B <sub>hex</sub>	<i>N</i>	Status	5



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the input assemblies with safety input data, combined safety input status, and individual safety output status.

**Table 6-6.22 Instance Numbers 270 -27D<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Input/V	Safety Discrete Input Point	3D <sub>hex</sub>	N	Safety Input Logical Value	7
Safety Input Status	Safety Discrete Input Group	3E <sub>hex</sub>	1	Status	5
Safety Output/V Status	Safety Discrete Output Point	3B <sub>hex</sub>	N	Status	5

### 6-6.8.3 Safety Data and Standard Data Open Assemblies

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the input assemblies with safety input data and standard input data.

**Table 6-6.23 Instance Numbers 281 -28D<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Input/V	Safety Discrete Input Point	3D <sub>hex</sub>	N	Safety Input Logical Value	7
Standard Input/V (standard data)	Safety Discrete Input Point	3D <sub>hex</sub>	N	Standard Input Value	10

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the input assemblies with safety input data and individual safety output monitor.

**Table 6-6.24 Instance Numbers 291 -29D<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Input/V	Safety Discrete Input Point	3D <sub>hex</sub>	N	Safety Input Logical Value	7
Safety Output/V Monitor (standard data)	Safety Discrete Output Point	3B <sub>hex</sub>	N	Output Port Value	4



**Safety Discrete I/O Device, Type: 23<sub>Hex</sub>**

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the input assemblies with safety input data, combined safety input status, and individual safety output monitor.

**Table 6-6.25 Instance Numbers 2A1 -2AD<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Input/V	Safety Discrete Input Point	3D <sub>hex</sub>	N	Safety Input Logical Value	7
Safety Input/V Status	Safety Discrete Input Group	3E <sub>hex</sub>	1	Status	5
Safety Output/V Monitor (standard data)	Safety Discrete Output Point	3B <sub>hex</sub>	N	Output Port Value	4

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the input assemblies with safety input data, individual safety input status, individual safety output status, and individual safety output monitor.

**Table 6-6.26 Instance Numbers 2B1 -2BD<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Input/V	Safety Discrete Input Point	3D <sub>hex</sub>	N	Safety Input Logical Value	7
Safety Input/V Status	Safety Discrete Input Point	3D <sub>hex</sub>	N	Status	5
Safety Output/V Status	Safety Discrete Output Point	3B <sub>hex</sub>	N	Status	5
Safety Output/V Monitor (standard data)	Safety Discrete Output Point	3B <sub>hex</sub>	N	Output Port Value	4

The following table indicates the I/O assembly Data attribute mapping for the Safety Discrete I/O device for the output assemblies with safety output data and standard output data.

**Table 6-6.27 Instance Numbers 2C1-2CD<sub>HEX</sub>**

Data Component Name	Class		Instance Number	Attribute	
	Name	Number		Name	Number
Safety Output N	Safety Discrete Output Point	3B <sub>hex</sub>	N	Safety Output Value	3
Safety Output N	Discrete Output Point	09 <sub>hex</sub>	N	Value	3



## 6-7 Safety Analog I/O Device

### Device Type: 2A hex

A Safety Analog I/O Device type interfaces to one or more Safety I/O devices that do not have network capabilities. Examples include Safety analog temperature and pressure sensors and valve actuators. A Safety Analog I/O device provides CIP Safety network communications to a safety controller and exchanges safety input and output data with safety analog sensors and actuators.

### 6-7.1 Object Model

The Object Model in Figure 6-7.1 represents a Safety Analog I/O Device. The Table below includes:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class
- **Note:** The Safety Analog Output Point, Safety Analog Output Group and Safety Dual Channel Output objects are included in higher level discussions of this profile for conceptual completeness. They are identified as future extensions to this profile. Details of these objects will be added as needed.

**Table 6-7.1 Objects Present in a Safety Analog I/O Device**

Object Class	Option/Required	# of Instances
Identity	Required	1
Message Router	Required	1
Link Specific Object(s)	Required	##
Connection	Required for DeviceNet Safety	#
Connection Manager	Required for EtherNet/IP Safety	1
Safety Validator	Required	#
Safety Supervisor	Required	1
Assembly	Required	*
Analog Input Point (AIP)	Optional <sup>3</sup>	*
Safety Analog Input Point (SAIP)	Conditional <sup>1</sup>	*
Analog Output Point (AOP)	Optional <sup>4</sup>	*
Safety Analog Output Point (SAOP)	Conditional <sup>2</sup>	*
Safety Dual Channel Analog Output (SDCAO)	Conditional <sup>5</sup>	*
Safety Dual Channel Analog Input (SDCAI)	Conditional <sup>5</sup>	*
Analog Input Group (AIG)	Optional	1
Safety Analog Input Group (SAIG)	Optional	1
Analog Output Group (AOG)	Optional	1
Safety Analog Output Group (SAOG)	Optional	1

\* = # of instances depends on the level of I/O supported by the product.

# = Depends on the level of safety IO connections supported by the product.

## = Depends on the number of network interfaces supported on the product.

1 = Required for Safety Input Functions

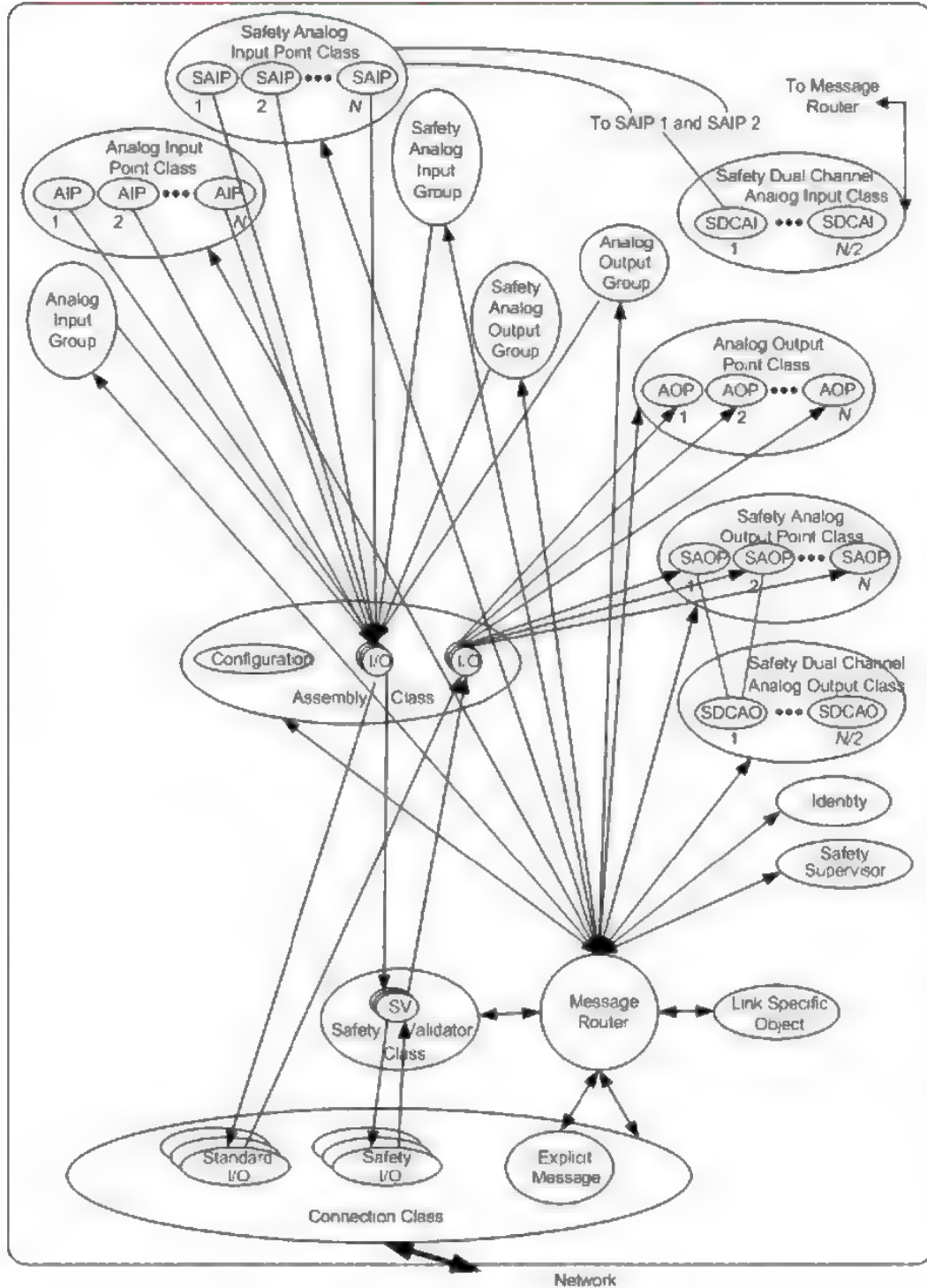
2 = Required for Safety Output Functions



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Object Class	Option/Required	# of Instances
3	= Optional for Standard Input Functions	
4	= Optional for Standard Output Functions	
5	= Required for Safety Input/Output Dual Channel Functions	

Figure 6-7.1 Object Model for Safety Analog I/O Device





## 6-7.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-7.2 Object Affect on Behavior**

Object	Effect on Behavior
Identity	No effect
Message Router	No effect
Link Specific Object	Configures communication link attributes
Connection (Manager) Object	Contains the logical ports into and out of the device
Assembly	Defines structure of the data available to standard and safety I/O connections and used to configure the device.
Safety Supervisor	Implements the Safety Network Configuration Tool Interface
Safety Validator	Performs the high integrity functions required for CIP Safety connections
Analog Input Point (AIP)	Defines behavior of the standard Analog Input Points for this device
Safety Analog Input Point (SAIP)	Defines behavior of the safety Analog Input Points for this device
Analog Output Point (AOP)	Defines behavior of the standard Analog Output Points for this device
Safety Analog Output Point (SAOP)	Defines behavior of the safety Analog Output Points for this device
Safety Dual Channel Analog Input	Defines behavior of the Safety Dual Channel Analog Inputs for this device
Safety Dual Channel Analog Output	Defines behavior of the Safety Dual Channel Analog Outputs for this device
Analog Input Group	Stores the grouped attributes of the Analog Input Points
Safety Analog Input Group	Stores the grouped attributes of the Safety Analog Input Points
Analog Output Group	Stores the grouped attributes of the Analog Output Points
Safety Analog Output Group	Stores the grouped attributes of the Safety Analog Output Points

### 6-7.2.1 Safety Supervisor Requirements

The safety supervisor definition contains a number of “profile dependent” functions. This section defines what the required behavior shall be for the Safety Analog I/O device profile.

**Table 6-7.3 Safety Supervisor Implementation Level**

Implementation Level	Profile Requirement
Baseline functionality	Required
SNCT functionality	Optional



**Table 6-7.4 Safety Supervisor Profile-dependent State Event Behavior**

Event	"IDLE" State Behavior	"Executing" State Behavior	Comments
Safety Connection Failed/Closed	Remain in IDLE	If any standard or safety I/O connection still open, remain in EXECUTING, Else, Transition to IDLE	In this profile, device is in IDLE unless at least one standard or Safety I/O connection is established
Standard or Safety I/O Connection established	Transition to EXECUTING	Remain in EXECUTING state	
Type 1 Safety Open	Configure device, transition to EXECUTING	Drop standard and safety I/O connections, configure device, return to EXECUTING	
Safety I/O Connection Deleted	Not supported	Not supported	Connection deletion not supported
Safety Supervisor Mode Change	Error response' "Service Not Supported"	Error response' "Service Not Supported"	No mode change defined for this profile

### 6-7.3 Defining Object Interfaces

The objects in this device have the interfaces listed in the following table.

**Table 6-7.5 Object Interfaces**

Object	Interface
Identity	Message Router
Message Router	Explicit Messaging Connection Instance
DeviceNet	Message Router
Connection	Message Router
Connection Manager	Message Router
Assembly	I/O Connection (for standard connections) or Safety Validator (for Safety I/O connections) or Message Router
Safety Supervisor	Message Router
Safety Validator	Message Router or Safety I/O Connection
Analog Input Point (AIP)	Message Router or Assembly Object
Safety Analog Input Point (SAIP)	Message Router or Assembly Object
Safety Dual Channel Analog Input (SDCAI)	Message Router or Assembly Object
Safety Dual Channel Analog Output (SDCAOI)	Message Router or Assembly Object
Analog Output Point (AOP)	Message Router or Assembly Object
Safety Analog Output Point (SAOP)	Message Router or Assembly Object
Safety Dual Channel Output	Message Router or Assembly Object
Analog Input Group	Message Router
Safety Analog Input Group	Message Router
Analog Output Group	Message Router
Safety Analog Output Group	Message Router



**6-7.4 Relationship between AIP and SAIP**

A module using the Safety Analog I/O profile may support either the SAIP, AIP, or both. An instance of the SAIP can be used to access inputs via safety evaluated data, or to access inputs as standard inputs without safety evaluation. An Instance of the AIP can be used as standard Analog input data which is equivalent to the SAIP input data without safety evaluation (Input Channel Mode set to Standard).

**6-7.5 Relationship between AOP and SAOP**

A module using the Safety Analog I/O profile may support either the SAOP, AOP, or both. An instance of the SAOP can be used to drive outputs in a high integrity manner, or to drive outputs as standard outputs with the high integrity validation disabled. An Instance of the AOP can be used as standard Analog output data which is equivalent to the SAOP output data without safety evaluation.

**6-7.6 I/O Assembly Instances**

The I/O Assemblies for the Safety Analog I/O Device may be classified into the following types.

**Table 6-7.6 I/O Assembly Object Instances for the Safety Analog I/O Device**

Input or Output	Data Type
Input	Standard Data Only
Input	Safety Data Only
Input	Safety and Standard Data
Output	Standard Data Only
Output	Safety Data Only
Output	Safety and Standard Data

The assembly instance definitions will differentiate safety data from standard data. Assembly instances that contain Safety Data shall be implemented in a high integrity manner.

When an input assembly can be accessed by both a Safety I/O connection and/or a Standard I/O connection, the same instance number will be used by either I/O connection type.

When an output assembly can be accessed by a Safety I/O connection or a Standard I/O connection, the same instance number will be used by either of the I/O connection types. Only one I/O connection can be made to an output assembly, or object, at any given time.

This profile will define Safety and Standard Data assemblies that are mapped to the SAIP that provide functionality comparable to AIP, but don't require the support of or the additional configuration of the AIP. These assemblies will be defined in the 180 – 2FF<sub>hex</sub> instance range.

As denoted in the table below instance number ranges 01 63<sub>hex</sub> and 100-17F<sub>hex</sub> of this profile will be reserved for definition by a General Purpose Analog I/O Profile. Those assembly definitions and mapping will not be repeated in this profile.



**Table 6-7.7 Instance Number Ranges**

Instance Range	Assembly Instance ID Range from Assembly Object Table 6.A	Qty	Data Type	Profile that defines the Assembly Instance	Typical Uses
01-63 <sub>hex</sub>	Open (static assemblies defined in device profile)	99	Standard Data Only	General Purpose Analog I/O	AIP or AOP based I/O data
64-C7 <sub>hex</sub>	Vendor Specific static assemblies and dynamic assemblies	100	Vendor Specific	Vendor Specific	AIP or AOP based I/O data
C8-FF <sub>hex</sub>	Reserved by CIP for future use	56			
100-17F <sub>hex</sub>	Open (static assemblies defined in device profile)	128	Standard Data Only	General Purpose Analog I/O	AIP or AOP based I/O data
180-1FF <sub>hex</sub>	Open (static assemblies defined in device profile)	128	16 bit Safety and Standard Data	Safety Analog I/O (this profile)	SAIP or SAOP based safety and standard I/O data
200-2FF <sub>hex</sub>	Open (static assemblies defined in device profile)	192	32 bit Safety and Standard Data	Safety Analog I/O (this profile)	SAIP or SAOP based safety and standard I/O data
300-4FF <sub>hex</sub>	Vendor Specific static assemblies and dynamic assemblies	512	Vendor Specific	Vendor Specific	
500-FFFF <sub>hex</sub>	Reserved by CIP for future use	64,256	-	-	

As denoted in the table above instance number range 180-2FF<sub>hex</sub> will be defined in this profile

### 6-7.6.1 Safety and Standard Data Open Assemblies

The following safety and standard data assemblies will be defined to allow access to the Safety I/O points as standard I/O points without having to support the AIP object. The Standard Input(n) data in these assemblies maps to the *Safety Analog Input Value* attribute of the SAIP. It is left to the application to configure and monitor the Safety versus Standard behavior of the SAIP. The Analog Input (n) data maps to the *Value* attribute of the AIP.

This **Individual Status** used in the following tables refers to the value of the SAIP attribute *Input Point Status*, which is the status of the individual analog channel.

The Safety Analog Input Point Object specifies that 8 bit, 16 bit, 32 bit and 64 bit data types may be implemented in that object. The initial revision of this profile defines assemblies for the 16 bit and 32 bit data types. Assemblies for other data types may be added to this profile in the future, or implemented as vendor specific assemblies.



**Table 6-7.8 Safety Data Open INT type Input Assembly Instances**

Instance Number	INT Type	Safety I/O	Individual Status
		Number of Safety Inputs	Input Point Status (n)
180 <sub>hex</sub>	In	1	
181 <sub>hex</sub>	In	2	
182 <sub>hex</sub>	In	4	
183 <sub>hex</sub>	In	6	
184 <sub>hex</sub>	In	8	
185 <sub>hex</sub>	In	10	
186 <sub>hex</sub>	In	12	
187 <sub>hex</sub>	In	14	
188 <sub>hex</sub>	In	16	
189 <sub>hex</sub>	In	18	
18A <sub>hex</sub>	In	20	
18B <sub>hex</sub>	In	24	
18C <sub>hex</sub>	In	28	
18D <sub>hex</sub>	In	32	
190 <sub>hex</sub>	In	1	1
191 <sub>hex</sub>	In	2	2
192 <sub>hex</sub>	In	4	4
193 <sub>hex</sub>	In	6	6
194 <sub>hex</sub>	In	8	8
195 <sub>hex</sub>	In	10	10
196 <sub>hex</sub>	In	12	12
197 <sub>hex</sub>	In	14	14
198 <sub>hex</sub>	In	16	16
199 <sub>hex</sub>	In	18	18
19A <sub>hex</sub>	In	20	20
19B <sub>hex</sub>	In	24	24
19C <sub>hex</sub>	In	28	28
19D <sub>hex</sub>	In	32	32



**Table 6-7.9 Safety Data Open INT type Output Assembly Instances**

Instance Number	INT Type	Safety I/O	Individual Status
		Number of Safety Outputs	Input Point Status (n)
1B0 <sub>hex</sub>	Out	1	
1B1 <sub>hex</sub>	Out	2	
1B2 <sub>hex</sub>	Out	4	
1B3 <sub>hex</sub>	Out	6	
1B4 <sub>hex</sub>	Out	8	
1B5 <sub>hex</sub>	Out	10	
1B6 <sub>hex</sub>	Out	12	
1B7 <sub>hex</sub>	Out	14	
1B8 <sub>hex</sub>	Out	16	
1B9 <sub>hex</sub>	Out	18	
1BA <sub>hex</sub>	Out	20	
1BB <sub>hex</sub>	Out	24	
1BC <sub>hex</sub>	Out	28	
1BD <sub>hex</sub>	Out	32	



Table 6-7.10 Safety Data Open UINT type Input Assembly Instances

Instance Number	UINT Type	Safety I/O	Individual Status
		Number of Safety Inputs	Input Point Status (n)
1C0 <sub>hex</sub>	In	1	
1C1 <sub>hex</sub>	In	2	
1C2 <sub>hex</sub>	In	4	
1C3 <sub>hex</sub>	In	6	
1C4 <sub>hex</sub>	In	8	
1C5 <sub>hex</sub>	In	10	
1C6 <sub>hex</sub>	In	12	
1C7 <sub>hex</sub>	In	14	
1C8 <sub>hex</sub>	In	16	
1C9 <sub>hex</sub>	In	18	
1CA <sub>hex</sub>	In	20	
1CB <sub>hex</sub>	In	24	
1CC <sub>hex</sub>	In	28	
1CD <sub>hex</sub>	In	32	
1D0 <sub>hex</sub>	In	1	1
1D1 <sub>hex</sub>	In	2	2
1D2 <sub>hex</sub>	In	4	4
1D3 <sub>hex</sub>	In	6	6
1D4 <sub>hex</sub>	In	8	8
1D5 <sub>hex</sub>	In	10	10
1D6 <sub>hex</sub>	In	12	12
1D7 <sub>hex</sub>	In	14	14
1D8 <sub>hex</sub>	In	16	16
1D9 <sub>hex</sub>	In	18	18
1DA <sub>hex</sub>	In	20	20
1DB <sub>hex</sub>	In	24	24
1DC <sub>hex</sub>	In	28	28
1DD <sub>hex</sub>	In	32	32



**Table 6-7.11 Safety Data Open UINT type Output Assembly Instances**

Instance Number	UINT Type	Safety I/O	Individual Status
		Number of Safety Outputs	Input Point Status (n)
1F0 <sub>hex</sub>	Out	1	
1F1 <sub>hex</sub>	Out	2	
1F2 <sub>hex</sub>	Out	4	
1F3 <sub>hex</sub>	Out	6	
1F4 <sub>hex</sub>	Out	8	
1F5 <sub>hex</sub>	Out	10	
1F6 <sub>hex</sub>	Out	12	
1F7 <sub>hex</sub>	Out	14	
1F8 <sub>hex</sub>	Out	16	
1F9 <sub>hex</sub>	Out	18	
1FA <sub>hex</sub>	Out	20	
1FB <sub>hex</sub>	Out	24	
1FC <sub>hex</sub>	Out	28	
1FD <sub>hex</sub>	Out	32	



**Table 6-7.12 Safety Data Open DINT type Assembly Instances**

Instance Number	DINT Type	Safety I/O	Individual Status
		Number of Safety Inputs	Input Point Status (n)
200 <sub>hex</sub>	In	1	
201 <sub>hex</sub>	In	2	
202 <sub>hex</sub>	In	4	
203 <sub>hex</sub>	In	6	
204 <sub>hex</sub>	In	8	
205 <sub>hex</sub>	In	10	
206 <sub>hex</sub>	In	12	
207 <sub>hex</sub>	In	14	
208 <sub>hex</sub>	In	16	
209 <sub>hex</sub>	In	18	
20A <sub>hex</sub>	In	20	
20B <sub>hex</sub>	In	24	
20C <sub>hex</sub>	In	28	
20D <sub>hex</sub>	In	32	
210 <sub>hex</sub>	In	1	1
211 <sub>hex</sub>	In	2	2
212 <sub>hex</sub>	In	4	4
213 <sub>hex</sub>	In	6	6
214 <sub>hex</sub>	In	8	8
215 <sub>hex</sub>	In	10	10
216 <sub>hex</sub>	In	12	12
217 <sub>hex</sub>	In	14	14
218 <sub>hex</sub>	In	16	16
219 <sub>hex</sub>	In	18	18
21A <sub>hex</sub>	In	20	20
21B <sub>hex</sub>	In	24	24
21C <sub>hex</sub>	In	28	28
21D <sub>hex</sub>	In	32	32



**Table 6-7.13 Safety Data Open DINT type Output Assembly Instances**

Instance Number	DINT Type	Safety I/O	Individual Status
		Number of Safety Outputs	Input Point Status (n)
230 <sub>hex</sub>	Out	1	
231 <sub>hex</sub>	Out	2	
232 <sub>hex</sub>	Out	4	
233 <sub>hex</sub>	Out	6	
234 <sub>hex</sub>	Out	8	
235 <sub>hex</sub>	Out	10	
236 <sub>hex</sub>	Out	12	
237 <sub>hex</sub>	Out	14	
238 <sub>hex</sub>	Out	16	
239 <sub>hex</sub>	Out	18	
23A <sub>hex</sub>	Out	20	
23B <sub>hex</sub>	Out	24	
23C <sub>hex</sub>	Out	28	
23D <sub>hex</sub>	Out	32	



**Table 6-7.14 Safety Data Open UDINT type Assembly Instances**

Instance Number	UDINT Type	Safety I/O	Individual Status
		Number of Safety Inputs	Input Point Status (n)
240 <sub>hex</sub>	In	1	
241 <sub>hex</sub>	In	2	
242 <sub>hex</sub>	In	4	
243 <sub>hex</sub>	In	6	
244 <sub>hex</sub>	In	8	
245 <sub>hex</sub>	In	10	
246 <sub>hex</sub>	In	12	
247 <sub>hex</sub>	In	14	
248 <sub>hex</sub>	In	16	
249 <sub>hex</sub>	In	18	
24A <sub>hex</sub>	In	20	
24B <sub>hex</sub>	In	24	
24C <sub>hex</sub>	In	28	
24D <sub>hex</sub>	In	32	
250 <sub>hex</sub>	In	1	1
251 <sub>hex</sub>	In	2	2
252 <sub>hex</sub>	In	4	4
253 <sub>hex</sub>	In	6	6
254 <sub>hex</sub>	In	8	8
255 <sub>hex</sub>	In	10	10
256 <sub>hex</sub>	In	12	12
257 <sub>hex</sub>	In	14	14
258 <sub>hex</sub>	In	16	16
259 <sub>hex</sub>	In	18	18
25A <sub>hex</sub>	In	20	20
25B <sub>hex</sub>	In	24	24
25C <sub>hex</sub>	In	28	28
25D <sub>hex</sub>	In	32	32



**Table 6-7.15 Safety Data Open UDINT type Output Assembly Instances**

Instance Number	UDINT Type	Safety I/O	Individual Status
		Number of Safety Outputs	Input Point Status (n)
270 <sub>hex</sub>	Out	1	
271 <sub>hex</sub>	Out	2	
272 <sub>hex</sub>	Out	4	
273 <sub>hex</sub>	Out	6	
274 <sub>hex</sub>	Out	8	
275 <sub>hex</sub>	Out	10	
276 <sub>hex</sub>	Out	12	
277 <sub>hex</sub>	Out	14	
278 <sub>hex</sub>	Out	16	
279 <sub>hex</sub>	Out	18	
27A <sub>hex</sub>	Out	20	
27B <sub>hex</sub>	Out	24	
27C <sub>hex</sub>	Out	28	
27D <sub>hex</sub>	Out	32	



**Table 6-7.16 Safety Data Open REAL type Input Assembly Instances**

Instance Number	REAL Type	Safety I/O	Individual Status
		Number of Safety Inputs	Input Point Status (n)
280 <sub>hex</sub>	In	1	
281 <sub>hex</sub>	In	2	
282 <sub>hex</sub>	In	4	
283 <sub>hex</sub>	In	6	
284 <sub>hex</sub>	In	8	
285 <sub>hex</sub>	In	10	
286 <sub>hex</sub>	In	12	
287 <sub>hex</sub>	In	14	
288 <sub>hex</sub>	In	16	
289 <sub>hex</sub>	In	18	
28A <sub>hex</sub>	In	20	
28B <sub>hex</sub>	In	24	
28C <sub>hex</sub>	In	28	
28D <sub>hex</sub>	In	32	
290 <sub>hex</sub>	In	1	1
291 <sub>hex</sub>	In	2	2
292 <sub>hex</sub>	In	4	4
293 <sub>hex</sub>	In	6	6
294 <sub>hex</sub>	In	8	8
295 <sub>hex</sub>	In	10	10
296 <sub>hex</sub>	In	12	12
297 <sub>hex</sub>	In	14	14
298 <sub>hex</sub>	In	16	16
299 <sub>hex</sub>	In	18	18
29A <sub>hex</sub>	In	20	20
29B <sub>hex</sub>	In	24	24
29C <sub>hex</sub>	In	28	28
29D <sub>hex</sub>	In	32	32



**Table 6-7.17 Safety Data Open REAL type Output Assembly Instances**

Instance Number	REAL Type	Safety I/O	Individual Status
		Number of Safety Outputs	Input Point Status (n)
2B0 <sub>hex</sub>	Out	1	
2B1 <sub>hex</sub>	Out	2	
2B2 <sub>hex</sub>	Out	4	
2B3 <sub>hex</sub>	Out	6	
2B4 <sub>hex</sub>	Out	8	
2B5 <sub>hex</sub>	Out	10	
2B6 <sub>hex</sub>	Out	12	
2B7 <sub>hex</sub>	Out	14	
2B8 <sub>hex</sub>	Out	16	
2B9 <sub>hex</sub>	Out	18	
2BA <sub>hex</sub>	Out	20	
2BB <sub>hex</sub>	Out	24	
2BC <sub>hex</sub>	Out	28	
2BD <sub>hex</sub>	Out	32	

The following tables define Status Data Only Assemblies.

**Table 6-7.18 Safety and Standard Status Data Assembly Instances**

Instance Number	Status	Individual Status	Fault Reason	Alarm Warning
		Number of Input Point Status	Fault Reason (n)	Alarm Warning (n)
2C0 <sub>hex</sub>	In	1		
2C1 <sub>hex</sub>	In	2		
2C2 <sub>hex</sub>	In	4		
2C3 <sub>hex</sub>	In	6		
2C4 <sub>hex</sub>	In	8		
2C5 <sub>hex</sub>	In	10		
2C6 <sub>hex</sub>	In	12		
2C7 <sub>hex</sub>	In	14		
2C8 <sub>hex</sub>	In	16		
2C9 <sub>hex</sub>	In	18		
2CA <sub>hex</sub>	In	20		
2CB <sub>hex</sub>	In	24		
2CC <sub>hex</sub>	In	28		
2CD <sub>hex</sub>	In	32		



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance Number	Status	Fault Reason	Alarm Warning
		Fault Reason (n)	Alarm Warning (n)
2D0 <sub>hex</sub>	In	1	
2D1 <sub>hex</sub>	In	2	
2D2 <sub>hex</sub>	In	4	
2D3 <sub>hex</sub>	In	6	
2D4 <sub>hex</sub>	In	8	
2D5 <sub>hex</sub>	In	10	
2D6 <sub>hex</sub>	In	12	
2D7 <sub>hex</sub>	In	14	
2D8 <sub>hex</sub>	In	16	
2D9 <sub>hex</sub>	In	18	
2DA <sub>hex</sub>	In	20	
2DB <sub>hex</sub>	In	24	
2DC <sub>hex</sub>	In	28	
2DD <sub>hex</sub>	In	32	

Instance Number	Status	Fault Reason	Alarm Warning
		Fault Reason (n)	Alarm Warning (n)
2E0 <sub>hex</sub>	In	1	1
2E1 <sub>hex</sub>	In	2	2
2E2 <sub>hex</sub>	In	4	4
2E3 <sub>hex</sub>	In	6	6
2E4 <sub>hex</sub>	In	8	8
2E5 <sub>hex</sub>	In	10	10
2E6 <sub>hex</sub>	In	12	12
2E7 <sub>hex</sub>	In	14	14
2E8 <sub>hex</sub>	In	16	16
2E9 <sub>hex</sub>	In	18	18
2EA <sub>hex</sub>	In	20	20
2EB <sub>hex</sub>	In	24	24
2EC <sub>hex</sub>	In	28	28
2ED <sub>hex</sub>	In	32	32



Instance Number	Status	Individual Status	Individual Monitor		
		Number of Safety Outputs	Number of Output Monitors		
2F0 <sub>hex</sub>	In	1	1		
2F1 <sub>hex</sub>	In	2	2		
2F2 <sub>hex</sub>	In	4	4		
2F3 <sub>hex</sub>	In	6	6		
2F4 <sub>hex</sub>	In	8	8		
2F5 <sub>hex</sub>	In	10	10		
2F6 <sub>hex</sub>	In	12	12		
2F7 <sub>hex</sub>	In	14	14		
2F8 <sub>hex</sub>	In	16	16		
2F9 <sub>hex</sub>	In	18	18		
2FA <sub>hex</sub>	In	20	20		
2FB <sub>hex</sub>	In	24	24		
2FC <sub>hex</sub>	In	28	28		
2FD <sub>hex</sub>	In	32	32		

### 6-7.7 I/O Assembly Data Attribute Format

The following sections define the IO data assemblies supported by the Safety Analog IO device type profile. Since these assemblies transfer analog data, the assembly sizes are large for even a few IO points. In order to make the following sections more readable, an abbreviated presentation style was used. For assemblies with six or less IO points, each data element is explicitly specified. For all other assemblies, an ellipsis (...) is used in the tables to indicate that the pattern established for the first two IO points is continued for the next number of IO points as needed to fill out the assembly. Additionally, rows in the tables show the Input (*n*) IO size in INT, UNIT, DINT, UDINT or REAL types for each IO point. Therefore, each entry for a single IO point in the assembly definition tables represents either two or four bytes of data. The actual byte count for the assemblies is listed in the “Byte” column of the tables.

#### 6-7.7.1 Safety and Standard Data Open Assemblies

The assemblies described in this section are defined to access the Safety I/O points as safe or standard I/O points. The Input (*n*) data in these assemblies maps to the *Safety Analog Input Value* attribute of the SAIP. Refer to section 6-7.8, Mapping IO Assembly Data Components. These assemblies may be accessed by either safety or standard IO connections.

Table 6-7.19 indicates that the INT and UINT data type Assemblies that have the same data content and the same number of IO points have the same format. The only difference is the Assembly Id. For example, the data format definitions for Assembly Id 180<sub>hex</sub> and Assembly Id 1C0<sub>hex</sub> are the same. Assembly Id 181 format and Assembly Id 1C1 format is the same, Assembly Id 191 format and Assembly Id 1D1 format is the same, and so forth for the rest of the Assembly Ids specified.



**Table 6-7.19 Safety and Standard Data 16 Bit Data Types and Assembly Instances**

Data Type	Number of IO Points													
	1	2	4	6	8	10	12	14	16	18	20	24	28	32
<b>Input Data only assemblies</b>														
INT	180	181	182	183	184	185	186	187	188	189	18A	18B	18C	18D
UINT	1C0	1C1	1C2	1C3	1C4	1C5	1C6	1C7	1C8	1C9	1CA	1CB	1CC	1CD
<b>Input Data and Status assemblies</b>														
INT	190	191	192	193	194	195	196	197	198	199	19A	19B	19C	19D
UINT	1D0	1D1	1D2	1D3	1D4	1D5	1D6	1D7	1D8	1D9	1DA	1DB	1DC	1DD
<b>Output Data only assemblies</b>														
INT	1B0	1B1	1B2	1B3	1B4	1B5	1B6	1B7	1B8	1B9	1BA	1BB	1BC	1BD
UINT	1F0	1F1	1F2	1F3	1F4	1F5	1F6	1F7	1F8	1F9	1FA	1FB	1FC	1FD

**Table 6-7.20 Safety and Standard Data 16 Bit Open Assembly Instances**

Instance	Byte	High Byte	Low Byte
180 <sub>hex</sub> 1C0 <sub>hex</sub>	0,1	Input 1	Input 1

Instance	Byte	High Byte	Low Byte
181 <sub>hex</sub>	0,1	Input 1	Input 1
1C1 <sub>hex</sub>	2,3	Input 2	Input 2

Instance	Byte	High Byte	Low Byte
182 <sub>hex</sub>	0,1	Input 1	Input 1
1C2 <sub>hex</sub>	2,3	Input 2	Input 2
	4,5	Input 3	Input 3
	6,7	Input 4	Input 4

Instance	Byte	High Byte	Low Byte
183 <sub>hex</sub>	0,1	Input 1	Input 1
1C3 <sub>hex</sub>	2,3	Input 2	Input 2
	4,5	Input 3	Input 3
	6,7	Input 4	Input 4
	8,9	Input 5	Input 5
	10,11	Input 6	Input 6

Instance	Byte	High Byte	Low Byte
184 <sub>hex</sub>	0,1	Input 1	Input 1
1C4 <sub>hex</sub>	2,3	Input 2	Input 2
	...	...	...
	12,13	Input 7	Input 7
	14,15	Input 8	Input 8



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte	Low Byte
185 <sub>hex</sub>	0,1	Input 1	Input 1
1C5 <sub>hex</sub>	2,3	Input 2	Input 2
	16,17	Input 9	Input 9
	18,19	Input 10	Input 10

Instance	Byte	High Byte	Low Byte
186 <sub>hex</sub>	0,1	Input 1	Input 1
1C6 <sub>hex</sub>	2,3	Input 2	Input 2
	...	...	...
	20,21	Input 11	Input 11
	22,23	Input 12	Input 12

Instance	Byte	High Byte	Low Byte
187 <sub>hex</sub>	0,1	Input 1	Input 1
1C7 <sub>hex</sub>	2,3	Input 2	Input 2
	24,25	Input 13	Input 13
	26,27	Input 14	Input 14

Instance	Byte	High Byte	Low Byte
188 <sub>hex</sub>	0,1	Input 1	Input 1
1C8 <sub>hex</sub>	2,3	Input 2	Input 2
	28,29	Input 15	Input 15
	30,31	Input 16	Input 16

Instance	Byte	High Byte	Low Byte
189 <sub>hex</sub>	0,1	Input 1	Input 1
1C9 <sub>hex</sub>	2,3	Input 2	Input 2
	...	...	...
	32,33	Input 17	Input 17
	34,35	Input 18	Input 18

Instance	Byte	High Byte	Low Byte
18A <sub>hex</sub>	0,1	Input 1	Input 1
1CA <sub>hex</sub>	2,3	Input 2	Input 2
	36,37	Input 19	Input 19
	38,39	Input 20	Input 20



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte	Low Byte
18B <sub>hex</sub>	0,1	Input 1	Input 1
1CB <sub>hex</sub>	2,3	Input 2	Input 2
	44,45	Input 23	Input 23
	46,47	Input 24	Input 24

Instance	Byte	High Byte	Low Byte
18C <sub>hex</sub>	0,1	Input 1	Input 1
1CC <sub>hex</sub>	2,3	Input 2	Input 2
	...	...	...
	52,53	Input 27	Input 27
	54,55	Input 28	Input 28

Instance	Byte	High Byte	Low Byte
18D <sub>hex</sub>	0,1	Input 1	Input 1
1CD <sub>hex</sub>	2,3	Input 2	Input 2
	60,61	Input 31	Input 31
	62,63	Input 32	Input 32

Table 6-7.21 Safety and Standard Data 16 bit Open Assembly Instances with Status

Instance	Byte	High Byte				Low Byte			
190 <sub>hex</sub>	0,1	Input 1				Input 1			
1D0 <sub>hex</sub>		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	2	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Input1 Status

Instance	Byte	High Byte				Low Byte			
191 <sub>hex</sub>	0,1	Input 1				Input 1			
1D1 <sub>hex</sub>	2,3	Input 2				Input 2			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	4	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Input2 Status	Input1 Status

Instance	Byte	High Byte				Low Byte			
192 <sub>hex</sub>	0,1	Input 1				Input 1			
1D2 <sub>hex</sub>	2,3	Input 2				Input 2			
	4,5	Input 3				Input 3			
	6,7	Input 4				Input 4			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	8	Reserved	Reserved	Reserved	Reserved	Input4 Status	Input3 Status	Input2 Status	Input1 Status



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte				Low Byte			
193 <sub>hex</sub> 1D3 <sub>hex</sub>	0,1	Input 1				Input 1			
	2,3	Input 2				Input 2			
	4,5	Input 3				Input 3			
	6,7	Input 4				Input 4			
	8,9	Input 5				Input 5			
	10,11	Input 6				Input 6			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	12	Reserved	Reserved	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status

Instance	Byte	High Byte				Low Byte			
194 <sub>hex</sub> 1D4 <sub>hex</sub>	0,1	Input 1				Input 1			
	2,3	Input 2				Input 2			
						..			
	13,14	Input 7				Input 7			
	14,15	Input 8				Input 8			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	16	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status

Instance	Byte	High Byte				Low Byte			
195 <sub>hex</sub> 1D5 <sub>hex</sub>	0,1	Input 1				Input 1			
	2,3	Input 2				Input 2			
	16,17	Input 9				Input 9			
	18,19	Input 10				Input 10			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	20	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	21	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Input10 Status	Input9 Status

Instance	Byte	High Byte				Low Byte			
196 <sub>hex</sub> 1D6 <sub>hex</sub>	0,1	Input 1				Input 1			
	2,3	Input 2				Input 2			
	...	...				...			
	20,21	Input 11				Input 11			
	22,23	Input 12				Input 12			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	24	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	25	Reserved	Reserved	Reserved	Reserved	Input12 Status	Input11 Status	Input10 Status	Input9 Status



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte				Low Byte			
197 <sub>hex</sub> 1D7 <sub>hex</sub>	0,1	Input 1				Input 1			
	2,3	Input 2				Input 2			
	24,25	Input 13				Input 13			
	26,27	Input 14				Input 14			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	28	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	29	Reserved	Reserved	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status

Instance	Byte	High Byte				Low Byte			
198 <sub>hex</sub> 1D8 <sub>hex</sub>	0,1	Input 1				Input 1			
	2,3	Input 2				Input 2			
						...			
	28,29	Input 15				Input 15			
	30,31	Input 16				Input 16			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	32	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	33	Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status

Instance	Byte	High Byte				Low Byte			
199 <sub>hex</sub> 1D9 <sub>hex</sub>	0,1	Input 1				Input 1			
	2,3	Input 2				Input 2			
						...			
	32,33	Input 17				Input 17			
	34,35	Input 18				Input 18			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	36	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	37	Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
	38	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Input18 Status	Input17 Status



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte				Low Byte			
19A <sub>hex</sub> 1DA <sub>hex</sub>	0,1	Input 1				Input 1			
	2,3	Input 2				Input 2			
	36,37	Input 19				Input 19			
	38,39	Input 20				Input 20			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	40	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	41	Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
	42	Reserved	Reserved	Reserved	Reserved	Input20 Status	Input19 Status	Input18 Status	Input17 Status

Instance	Byte	High Byte				Low Byte			
19B <sub>hex</sub> 1DB <sub>hex</sub>	0,1	Input 1				Input 1			
	2,3	Input 2				Input 2			
	44,45	Input 23				Input 23			
	46,47	Input 24				Input 24			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	48	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	49	Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
	50	Input24 Status	Input23 Status	Input22 Status	Input21 Status	Input20 Status	Input19 Status	Input18 Status	Input17 Status

Instance	Byte	High Byte				Low Byte			
19C <sub>hex</sub> 1DC <sub>hex</sub>	0,1	Input 1				Input 1			
	2,3	Input 2				Input 2			
	52,53	Input 27				Input 27			
	54,55	Input 28				Input 28			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	56	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	57	Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
	58	Input24 Status	Input23 Status	Input22 Status	Input21 Status	Input20 Status	Input19 Status	Input18 Status	Input17 Status
	59	Reserved	Reserved	Reserved	Reserved	Input28 Status	Input27 Status	Input26 Status	Input25 Status



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte				Low Byte			
19D <sub>hex</sub>	0,1	Input 1				Input 1			
1DD <sub>hex</sub>	2,3	Input 2				Input 2			
	60,61	Input 31				Input 31			
	62,63	Input 32				Input 32			
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	64	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	65	Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
	66	Input24 Status	Input23 Status	Input22 Status	Input21 Status	Input20 Status	Input19 Status	Input18 Status	Input17 Status
	67	Input32 Status	Input31 Status	Input30 Status	Input29 Status	Input28 Status	Input27 Status	Input26 Status	Input25 Status

Table 6-7.22 Safety and Standard Data 16 bit Open Output Assembly Instances

Instance	Byte	High Byte	Low Byte
1B0 <sub>hex</sub> 1F0 <sub>hex</sub>	0,1	Output 1	Output 1

Instance	Byte	High Byte	Low Byte
1B1 <sub>hex</sub>	0,1	Output 1	Output 1
1F1 <sub>hex</sub>	2,3	Output 2	Output 2

Instance	Byte	High Byte	Low Byte
1B2 <sub>hex</sub>	0,1	Output 1	Output 1
1F2 <sub>hex</sub>	2,3	Output 2	Output 2
	4,5	Output 3	Output 3
	6,7	Output 4	Output 4

Instance	Byte	High Byte	Low Byte
1B3 <sub>hex</sub>	0,1	Output 1	Output 1
1F3 <sub>hex</sub>	2,3	Output 2	Output 2
	4,5	Output 3	Output 3
	6,7	Output 4	Output 4
	8,9	Output 5	Output 5
	10,11	Output 6	Output 6

Instance	Byte	High Byte	Low Byte
1B4 <sub>hex</sub>	0,1	Output 1	Output 1
1F4 <sub>hex</sub>	2,3	Output 2	Output 2
	12,13	Output 7	Output 7
	14,15	Output 8	Output 8



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte	Low Byte
1B5 <sub>hex</sub> 1F5 <sub>hex</sub>	0,1	Output 1	Output 1
	2,3	Output 2	Output 2
	16,17	Output 9	Output 9
	18,19	Output 10	Output 10

Instance	Byte	High Byte	Low Byte
1B6 <sub>hex</sub> 1F6 <sub>hex</sub>	0,1	Output 1	Output 1
	2,3	Output 2	Output 2
	...	...	...
	20,21	Output 11	Output 11
	22,23	Output 12	Output 12

Instance	Byte	High Byte	Low Byte
1B7 <sub>hex</sub> 1F7 <sub>hex</sub>	0,1	Output 1	Output 1
	2,3	Output 2	Output 2
	24,25	Output 13	Output 13
	26,27	Output 14	Output 14

Instance	Byte	High Byte	Low Byte
1B8 <sub>hex</sub> 1F8 <sub>hex</sub>	0,1	Output 1	Output 1
	2,3	Output 2	Output 2
	28,29	Output 15	Output 15
	30,31	Output 16	Output 16

Instance	Byte	High Byte	Low Byte
1B9 <sub>hex</sub> 1F9 <sub>hex</sub>	0,1	Output 1	Output 1
	2,3	Output 2	Output 2
	...	...	...
	32,33	Output 17	Output 17
	34,35	Output 18	Output 18

Instance	Byte	High Byte	Low Byte
1BA <sub>hex</sub> 1FA <sub>hex</sub>	0,1	Output 1	Output 1
	2,3	Output 2	Output 2
	36,37	Output 19	Output 19
	38,39	Output 20	Output 20



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte	Low Byte
1BB <sub>hex</sub>	0,1	Output 1	Output 1
1FB <sub>hex</sub>	2,3	Output 2	Output 2
	44,45	Output 23	Output 23
	46,47	Output 24	Output 24

Instance	Byte	High Byte	Low Byte
1BC <sub>hex</sub>	0,1	Output 1	Output 1
1FC <sub>hex</sub>	2,3	Output 2	Output 2
	...	...	...
	52,53	Output 27	Output 27
	54,55	Output 28	Output 28

Instance	Byte	High Byte	Low Byte
1BD <sub>hex</sub>	0,1	Output 1	Output 1
1FD <sub>hex</sub>	2,3	Output 2	Output 2
	60,61	Output 31	Output 31
	62,63	Output 32	Output 32

The following table indicates that the 32-bit data type Assemblies defined in this profile with the same data content and the same numbers of IO points have the same format. The only difference is the Assembly Id. For example, the data definitions for Assembly Id 200<sub>hex</sub> and Assembly Id 240<sub>hex</sub> and 280<sub>hex</sub> are the same. Assembly Id 201<sub>hex</sub> and Assembly Id 241<sub>hex</sub> and 281<sub>hex</sub> are the same, Assembly Id 211<sub>hex</sub> and Assembly Id 251<sub>hex</sub> and 291<sub>hex</sub> are the same, and so forth for the rest of the Assembly Ids specified.

Table 6-7.23 Safety and Standard Data 32 bit Data Types and Assembly Instances

Data Type	Number of IO Points													
	1	2	4	6	8	10	12	14	16	18	20	24	28	32
<b>Input Data only assemblies</b>														
DINT	200	201	202	203	204	205	206	207	208	209	20A	20B	20C	20D
UDINT	240	241	242	243	244	245	246	247	248	249	24A	24B	24C	24D
REAL	280	281	282	283	284	285	286	287	288	289	28A	28B	28C	28D
<b>Input Data and Status assemblies</b>														
DINT	210	211	212	213	214	215	216	217	218	219	21A	21B	21C	21D
UDINT	250	251	252	253	254	255	256	257	258	259	25A	25B	25C	25D
REAL	290	291	292	293	294	295	296	297	298	299	29A	29B	29C	29D
<b>Output Data only assemblies</b>														
DINT	230	231	232	233	234	235	236	237	238	239	23A	23B	23C	23D
UDINT	270	271	272	273	274	275	276	277	278	279	27A	27B	27C	27D
REAL	2B0	2B1	2B2	2B3	2B4	2B5	2B6	2B7	2B8	2B9	2BA	2BB	2BC	2BD



Table 6-7.24 Safety and Standard Data 32 bit Open Assembly Instances

Instance	Byte	High Byte	Byte	Byte	Low Byte
200 <sub>hex</sub> 240 <sub>hex</sub> 280 <sub>hex</sub>	0 - 3	Input 1	Input 1	Input 1	Input 1
201 <sub>hex</sub> 241 <sub>hex</sub> 281 <sub>hex</sub>	0 - 3 4 - 7	Input 1 Input 2	Input 1 Input 2	Input 1 Input 2	Input 1 Input 2
202 <sub>hex</sub> 242 <sub>hex</sub> 282 <sub>hex</sub>	0 - 3 4 - 7 8 - 11 12 - 15	Input 1 Input 2 Input 3 Input 4	Input 1 Input 2 Input 3 Input 4	Input 1 Input 2 Input 3 Input 4	Input 1 Input 2 Input 3 Input 4
203 <sub>hex</sub> 243 <sub>hex</sub> 283 <sub>hex</sub>	0 - 3 4 - 7 8 - 11 12 - 15 16 - 19 20 - 23	Input 1 Input 2 Input 3 Input 4 Input 5 Input 6	Input 1 Input 2 Input 3 Input 4 Input 5 Input 6	Input 1 Input 2 Input 3 Input 4 Input 5 Input 6	Input 1 Input 2 Input 3 Input 4 Input 5 Input 6
204 <sub>hex</sub> 244 <sub>hex</sub> 284 <sub>hex</sub>	0 - 3 4 - 7 ... 24 - 27 28 - 31	Input 1 Input 2 ... Input 7 Input 8	Input 1 Input 2 ... Input 7 Input 8	Input 1 Input 2 ... Input 7 Input 8	Input 1 Input 2 ... Input 7 Input 8
205 <sub>hex</sub> 245 <sub>hex</sub> 285 <sub>hex</sub>	0 - 3 4 - 7 ... 32 - 35 36 - 39	Input 1 Input 2 ... Input 9 Input 10	Input 1 Input 2 ... Input 9 Input 10	Input 1 Input 2 ... Input 9 Input 10	Input 1 Input 2 ... Input 9 Input 10
206 <sub>hex</sub> 246 <sub>hex</sub> 286 <sub>hex</sub>	0 - 3 4 - 7 ... 40 - 43 44 - 47	Input 1 Input 2 ... Input 11 Input 12	Input 1 Input 2 ... Input 11 Input 12	Input 1 Input 2 ... Input 11 Input 12	Input 1 Input 2 ... Input 11 Input 12



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte	Byte	Byte	Low Byte
207 <sub>hex</sub>	0 - 3	Input 1	Input 1	Input 1	Input 1
247 <sub>hex</sub>	4 - 7	Input 2	Input 2	Input 2	Input 2
287 <sub>hex</sub>					
	48 - 51	Input 13	Input 13	Input 13	Input 13
	52 - 55	Input 14	Input 14	Input 14	Input 14

Instance	Byte	High Byte	Byte	Byte	Low Byte
208 <sub>hex</sub>	0 - 3	Input 1	Input 1	Input 1	Input 1
248 <sub>hex</sub>	4 - 7	Input 2	Input 2	Input 2	Input 2
288 <sub>hex</sub>		...	...	...	...
	56 - 59	Input 15	Input 15	Input 15	Input 15
	60 - 63	Input 16	Input 16	Input 16	Input 16

Instance	Byte	High Byte	Byte	Byte	Low Byte
209 <sub>hex</sub>	0 - 3	Input 1	Input 1	Input 1	Input 1
249 <sub>hex</sub>	4 - 7	Input 2	Input 2	Input 2	Input 2
289 <sub>hex</sub>	...	...	...	...	...
	65 - 67	Input 17	Input 17	Input 17	Input 17
	68 - 71	Input 18	Input 18	Input 18	Input 18

Instance	Byte	High Byte	Byte	Byte	Low Byte
20A <sub>hex</sub>	0 - 3	Input 1	Input 1	Input 1	Input 1
24A <sub>hex</sub>	4 - 7	Input 2	Input 2	Input 2	Input 2
28A <sub>hex</sub>	...	...	...	...	...
	72 - 75	Input 19	Input 19	Input 19	Input 19
	76 - 79	Input 20	Input 20	Input 20	Input 20

Instance	Byte	High Byte	Byte	Byte	Low Byte
20B <sub>hex</sub>	0 - 3	Input 1	Input 1	Input 1	Input 1
24B <sub>hex</sub>	4 - 7	Input 2	Input 2	Input 2	Input 2
28B <sub>hex</sub>	...	...	...	...	...
	88 - 91	Input 23	Input 23	Input 23	Input 23
	92 - 95	Input 24	Input 24	Input 24	Input 24

Instance	Byte	High Byte	Byte	Byte	Low Byte
20C <sub>hex</sub>	0 - 3	Input 1	Input 1	Input 1	Input 1
24C <sub>hex</sub>	4 - 7	Input 2	Input 2	Input 2	Input 2
28C <sub>hex</sub>		..	...		..
	104 - 107	Input 27	Input 27	Input 27	Input 27
	108 - 111	Input 28	Input 28	Input 28	Input 28



Instance	Byte	High Byte	Byte	Byte	Low Byte
20D <sub>hex</sub>	0 - 3	Input 1	Input 1	Input 1	Input 1
24D <sub>hex</sub>	4 - 7	Input 2	Input 2	Input 2	Input 2
28D <sub>hex</sub>					
	120 - 123	Input 31	Input 31	Input 31	Input 31
	124 - 127	Input 32	Input 32	Input 32	Input 32

**Table 6-7.25 Safety and Standard Data 32 bit Open Assembly Instances with Status**

[illegible]

Instance	Byte	High Byte		Byte		Byte		Low Byte	
211 <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
251 <sub>hex</sub>	4 - 7	Input 2		Input 2		Input 2		Input 2	
291 <sub>hex</sub>		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	32	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Input2 Status	Input1 Status

Instance	Byte	High Byte		Byte		Byte		Low Byte	
212 <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
252 <sub>hex</sub>	4 - 7	Input 2		Input 2		Input 2		Input 2	
292 <sub>hex</sub>	8 - 11	Input 3		Input 3		Input 3		Input 3	
	12 - 15	Input 4		Input 4		Input 4		Input 4	
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	32	Reserved	Reserved	Reserved	Reserved	Input4 Status	Input3 Status	Input2 Status	Input1 Status

Instance	Byte	High Byte		Byte		Byte		Low Byte	
213 <sub>hex</sub> 253 <sub>hex</sub> 293 <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
	4 - 7	Input 2		Input 2		Input 2		Input 2	
	8 - 11	Input 3		Input 3		Input 3		Input 3	
	12 - 15	Input 4		Input 4		Input 4		Input 4	
	16 - 19	Input 5		Input 5		Input 5		Input 5	
	20 - 23	Input 6		Input 6		Input 6		Input 6	
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	32	Reserved	Reserved	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte		Byte		Byte		Low Byte	
214 <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
254 <sub>hex</sub>	4 - 7	Input 2		Input 2		Input 2		Input 2	
294 <sub>hex</sub>	...	...		...		...		...	
	24 - 27	Input 7		Input 7		Input 7		Input 7	
	28 - 31	Input 8		Input 8		Input 8		Input 8	
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	32	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status

Instance	Byte	High Byte		Byte		Byte		Low Byte	
215 <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
255 <sub>hex</sub>	4 - 7	Input 2		Input 2		Input 2		Input 2	
295 <sub>hex</sub>	...	...		...		...		...	
	32 - 35	Input 9		Input 9		Input 9		Input 9	
	36 - 39	Input 10		Input 10		Input 10		Input 10	
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	40	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	41	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Input10 Status	Input9 Status

Instance	Byte	High Byte		Byte		Byte		Low Byte	
216 <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
256 <sub>hex</sub>	4 - 7	Input 2		Input 2		Input 2		Input 2	
296 <sub>hex</sub>	...	...		...		...		...	
	40 - 43	Input 11		Input 11		Input 11		Input 11	
	44 - 47	Input 12		Input 12		Input 12		Input 12	
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	24	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	25	Reserved	Reserved	Reserved	Reserved	Input12 Status	Input11 Status	Input10 Status	Input9 Status

Instance	Byte	High Byte		Byte		Byte		Low Byte	
217 <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
257 <sub>hex</sub>	4 - 7	Input 2		Input 2		Input 2		Input 2	
297 <sub>hex</sub>	...	...		...		...		...	
	48 - 51	Input 13		Input 13		Input 13		Input 13	
	52 - 55	Input 14		Input 14		Input 14		Input 14	
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	56	Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	57	Reserved	Reserved	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte		Byte		Byte		Low Byte	
218 <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
258 <sub>hex</sub>	4 - 7	Input 2		Input 2		Input 2		Input 2	
298 <sub>hex</sub>									
	56 - 59	Input 15		Input 15		Input 15		Input 15	
	60 - 63	Input 16		Input 16		Input 16		Input 16	
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
64		Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
65		Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status

Instance	Byte	High Byte		Byte		Byte		Low Byte	
219 <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
259 <sub>hex</sub>	4 - 7	Input 2		Input 2		Input 2		Input 2	
299 <sub>hex</sub>	...	...		...		...		...	
	65 - 67	Input 17		Input 17		Input 17		Input 17	
	68 - 71	Input 18		Input 18		Input 18		Input 18	
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
72		Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
73		Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
74		Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Input18 Status	Input17 Status

Instance	Byte	High Byte		Byte		Byte		Low Byte	
21A <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
25A <sub>hex</sub>	4 - 7	Input 2		Input 2		Input 2		Input 2	
29A <sub>hex</sub>	...	...		...		...		...	
	72 - 75	Input 19		Input 19		Input 19		Input 19	
	76 - 79	Input 20		Input 20		Input 20		Input 20	
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
80		Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
81		Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
82		Reserved	Reserved	Reserved	Reserved	Input20 Status	Input19 Status	Input18 Status	Input17 Status



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte		Byte		Byte		Low Byte	
21B <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
25B <sub>hex</sub>	4 - 7	Input 2		Input 2		Input 2		Input 2	
29B <sub>hex</sub>	..	..							
	88 - 91	Input 23		Input 23		Input 23		Input 23	
	92 - 95	Input 24		Input 24		Input 24		Input 24	
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
96		Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
97		Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
98		Input24 Status	Input23 Status	Input22 Status	Input21 Status	Input20 Status	Input19 Status	Input18 Status	Input17 Status

Instance	Byte	High Byte		Byte		Byte		Low Byte	
21C <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
25C <sub>hex</sub>	4 - 7	Input 2		Input 2		Input 2		Input 2	
29C <sub>hex</sub>	...	..		...					
	104 - 107	Input 27		Input 27		Input 27		Input 27	
	108 - 111	Input 28		Input 28		Input 28		Input 28	
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
112		Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
113		Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
114		Input24 Status	Input23 Status	Input22 Status	Input21 Status	Input20 Status	Input19 Status	Input18 Status	Input17 Status
115		Reserved	Reserved	Reserved	Reserved	Input28 Status	Input27 Status	Input26 Status	Input25 Status

Instance	Byte	High Byte		Byte		Byte		Low Byte	
21D <sub>hex</sub>	0 - 3	Input 1		Input 1		Input 1		Input 1	
25D <sub>hex</sub>	4 - 7	Input 2		Input 2		Input 2		Input 2	
29D <sub>hex</sub>	...	..		...					
	120 - 123	Input 31		Input 31		Input 31		Input 31	
	124 - 127	Input 32		Input 32		Input 32		Input 32	
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
128		Input8 Status	Input7 Status	Input6 Status	Input5 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
129		Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
130		Input24 Status	Input23 Status	Input22 Status	Input21 Status	Input20 Status	Input19 Status	Input18 Status	Input17 Status
131		Input32 Status	Input31 Status	Input30 Status	Input29 Status	Input28 Status	Input27 Status	Input26 Status	Input25 Status



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Table 6-7.26 Safety and Standard Data 32 bit Open Output Assembly Instances

Instance	Byte	High Byte	Byte	Byte	Low Byte
230 <sub>hex</sub> 270 <sub>hex</sub> 2B0 <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1

Instance	Byte	High Byte	Byte	Byte	Low Byte
231 <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
271 <sub>hex</sub> 2B1 <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2

Instance	Byte	High Byte	Byte	Byte	Low Byte
232 <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
272 <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2
2B2 <sub>hex</sub>	8 - 11	Output 3	Output 3	Output 3	Output 3
	12 - 15	Output 4	Output 4	Output 4	Output 4

Instance	Byte	High Byte	Byte	Byte	Low Byte
233 <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
273 <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2
2B3 <sub>hex</sub>	8 - 11	Output 3	Output 3	Output 3	Output 3
	12 - 15	Output 4	Output 4	Output 4	Output 4
	16 - 19	Output 5	Output 5	Output 5	Output 5
	20 - 23	Output 6	Output 6	Output 6	Output 6

Instance	Byte	High Byte	Byte	Byte	Low Byte
234 <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
274 <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2
2B4 <sub>hex</sub>			...		..
	24 - 27	Output 7	Output 7	Output 7	Output 7
	28 - 31	Output 8	Output 8	Output 8	Output 8

Instance	Byte	High Byte	Byte	Byte	Low Byte
235 <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
275 <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2
2B5 <sub>hex</sub>	...	...	...	...	...
	32 - 35	Output 9	Output 9	Output 9	Output 9
	36 - 39	Output 10	Output 10	Output 10	Output 10

Instance	Byte	High Byte	Byte	Byte	Low Byte
236 <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
276 <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2
2B6 <sub>hex</sub>					..
	40 - 43	Output 11	Output 11	Output 11	Output 11
	44 - 47	Output 12	Output 12	Output 12	Output 12



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte	Byte	Byte	Low Byte
237 <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
277 <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2
2B7 <sub>hex</sub>	...	...	...	...	...
	48 - 51	Output 13	Output 13	Output 13	Output 13
	52 - 55	Output 14	Output 14	Output 14	Output 14

Instance	Byte	High Byte	Byte	Byte	Low Byte
238 <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
278 <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2
2B8 <sub>hex</sub>	...	...	...	...	...
	56 - 59	Output 15	Output 15	Output 15	Output 15
	60 - 63	Output 16	Output 16	Output 16	Output 16

Instance	Byte	High Byte	Byte	Byte	Low Byte
239 <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
279 <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2
2B9 <sub>hex</sub>	...	...	...	...	...
	65 - 67	Output 17	Output 17	Output 17	Output 17
	68 - 71	Output 18	Output 18	Output 18	Output 18

Instance	Byte	High Byte	Byte	Byte	Low Byte
23A <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
27A <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2
2BA <sub>hex</sub>	...	...	...	...	...
	72 - 75	Output 19	Output 19	Output 19	Output 19
	76 - 79	Output 20	Output 20	Output 20	Output 20

Instance	Byte	High Byte	Byte	Byte	Low Byte
23B <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
27B <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2
2BB <sub>hex</sub>	...	...	...	...	...
	88 - 91	Output 23	Output 23	Output 23	Output 23
	92 - 95	Output 24	Output 24	Output 24	Output 24

Instance	Byte	High Byte	Byte	Byte	Low Byte
23C <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
27C <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2
2BC <sub>hex</sub>	...	...	...	...	...
	104 - 107	Output 27	Output 27	Output 27	Output 27
	108 - 111	Output 28	Output 28	Output 28	Output 28



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	High Byte	Byte	Byte	Low Byte
23D <sub>hex</sub>	0 - 3	Output 1	Output 1	Output 1	Output 1
27D <sub>hex</sub>	4 - 7	Output 2	Output 2	Output 2	Output 2
2BD <sub>hex</sub>	..	..			
	120 - 123	Output 31	Output 31	Output 31	Output 31
	124 - 127	Output 32	Output 32	Output 32	Output 32

Table 6-7.27 Safety and Standard Data Open Assembly Instances with Status Only

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C0 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Input1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C1 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Input2 Status	Input1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C2 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Reserved	Input4 Status	Input3 Status	Input2 Status	Input1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C3 <sub>hex</sub>	0	Reserved	Reserved	Input6 Status	Input15 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C4 <sub>hex</sub>	0	Input8 Status	Input7 Status	Input6 Status	Input15 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C5 <sub>hex</sub>	0	Input8 Status	Input7 Status	Input6 Status	Input15 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	1	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Input10 Status	Input9 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C6 <sub>hex</sub>	0	Input8 Status	Input7 Status	Input6 Status	Input15 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	1	Reserved	Reserved	Reserved	Reserved	Input12 Status	Input11 Status	Input10 Status	Input9 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C7 <sub>hex</sub>	0	Input8 Status	Input7 Status	Input6 Status	Input15 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	1	Reserved	Reserved	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C8 <sub>hex</sub>	0	Input8 Status	Input7 Status	Input6 Status	Input15 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	1	Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2C9 <sub>hex</sub>	0	Input8 Status	Input7 Status	Input6 Status	Input15 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	1	Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
	2	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Input18 Status	Input7 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2CA <sub>hex</sub>	0	Input8 Status	Input7 Status	Input6 Status	Input15 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	1	Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
	2	Reserved	Reserved	Reserved	Reserved	Input20 Status	Input19 Status	Input18 Status	Input17 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2CB <sub>hex</sub>	0	Input8 Status	Input7 Status	Input6 Status	Input15 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	1	Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
	2	Input24 Status	Input23 Status	Input22 Status	Input21 Status	Input20 Status	Input19 Status	Input18 Status	Input17 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2CC <sub>hex</sub>	0	Input8 Status	Input7 Status	Input6 Status	Input15 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	1	Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
	2	Input24 Status	Input23 Status	Input22 Status	Input21 Status	Input20 Status	Input19 Status	Input18 Status	Input17 Status
	3	Reserved	Reserved	Reserved	Reserved	Input28 Status	Input27 Status	Input26 Status	Input25 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2CD <sub>hex</sub>	0	Input8 Status	Input7 Status	Input6 Status	Input15 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	1	Input16 Status	Input15 Status	Input14 Status	Input13 Status	Input12 Status	Input11 Status	Input10 Status	Input9 Status
	2	Input24 Status	Input23 Status	Input22 Status	Input21 Status	Input20 Status	Input19 Status	Input18 Status	Input17 Status
	3	Input32 Status	Input31 Status	Input30 Status	Input29 Status	Input28 Status	Input27 Status	Input26 Status	Input25 Status



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2D0 <sub>hex</sub>	0	Fault Reason 1							
2D1 <sub>hex</sub>	0	Fault Reason 1							
	1	Fault Reason 2							
2D2 <sub>hex</sub>	0	Fault Reason 1							
	3	Fault Reason 3							
2D3 <sub>hex</sub>	0	Fault Reason 1							
		..							
	5	Fault Reason 6							
2D4 <sub>hex</sub>	0	Input8 Status	Input7 Status	Input6 Status	Input15 Status	Input4 Status	Input3 Status	Input2 Status	Input1 Status
	0	Fault Reason 1							
	...	...							
	7	Fault Reason 8							
2D5 <sub>hex</sub>	0	Fault Reason 1							
	...	...							
	9	Fault Reason 10							
2D6 <sub>hex</sub>	0	Fault Reason 1							
	...	...							
	11	Fault Reason 12							
2D7 <sub>hex</sub>	0	Fault Reason 1							
	13	Fault Reason 14							
2D8 <sub>hex</sub>	0	Fault Reason 1							
	15	Fault Reason 16							



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2D9 <sub>hex</sub>	0	Fault Reason 1							
		-							
	17	Fault Reason 18							
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2DA <sub>hex</sub>	0	Fault Reason 1							
		-							
	19	Fault Reason 20							
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2DB <sub>hex</sub>	0	Fault Reason 1							
	...	...							
	23	Fault Reason 24							
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2DC <sub>hex</sub>	0	Fault Reason 1							
	...	...							
	27	Fault Reason 28							
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2DD <sub>hex</sub>	0	Fault Reason 1							
		-							
	31	Fault Reason 32							
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2E0 <sub>hex</sub>	0	Fault Reason 1							
	1	Alarm/Warning Reason 1							
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2E1 <sub>hex</sub>	0	Fault Reason 1							
	1	Fault Reason 2							
	2	Alarm/Warning Reason 1							
	3	Alarm/Warning Reason 2							
Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2E2 <sub>hex</sub>	0	Fault Reason 1							
		-							
	3	Fault Reason 6							
	4	Alarm/Warning Reason 1							
		...							
	7	Alarm/Warning Reason 4							



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2E3 <sub>hex</sub>	0	Fault Reason 1							
		...							
	5	Fault Reason 6							
	6	Alarm/Warning Reason 1							
		...							
	11	Alarm/Warning Reason 6							

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2E4 <sub>hex</sub>	0	Fault Reason 1							
		...							
	7	Fault Reason 8							
	8	Alarm/Warning Reason 1							
		...							
	15	Alarm/Warning Reason 8							

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2E5 <sub>hex</sub>	0	Fault Reason 1							
		...							
	9	Fault Reason 10							
	10	Alarm/Warning Reason 1							
		...							
	17	Alarm/Warning Reason 10							

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2E6 <sub>hex</sub>	0	Fault Reason 1							
		...							
	11	Fault Reason 12							
	12	Alarm/Warning Reason 1							
		...							
	21	Alarm/Warning Reason 12							

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2E7 <sub>hex</sub>	0	Fault Reason 1							
		...							
	13	Fault Reason 14							
	14	Alarm/Warning Reason 1							
		...							
	27	Alarm/Warning Reason 14							



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2E8 <sub>hex</sub>	0	Fault Reason 1							
		...							
	15	Fault Reason 16							
	16	Alarm/Warning Reason 1							
	31	Alarm/Warning Reason 16							

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2E9 <sub>hex</sub>	0	Fault Reason 1							
		...							
	17	Fault Reason 18							
	18	Alarm/Warning Reason 1							
	33	Alarm/Warning Reason 18							

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2EA <sub>hex</sub>	0	Fault Reason 1							
		...							
	19	Fault Reason 20							
	20	Alarm/Warning Reason 1							
	39	Alarm/Warning Reason 20							

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2EB <sub>hex</sub>	0	Fault Reason 1							
		...							
	23	Fault Reason 24							
	24	Alarm/Warning Reason 1							
	47	Alarm/Warning Reason 24							

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2EC <sub>hex</sub>	0	Fault Reason 1							
		...							
	27	Fault Reason 28							
	28	Alarm/Warning Reason 1							
	55	Alarm/Warning Reason 28							



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2ED <sub>hex</sub>	0	Fault Reason 1							
	31	Fault Reason 32							
	32	Alarm/Warning Reason 1							
		...							
63		Alarm/Warning Reason 32							

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2F0 <sub>hex</sub>	0	Reserved	Reserved	Reserved	Output1 Monitor	Reserved	Reserved	Reserved	Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2F1 <sub>hex</sub>	0	Reserved	Reserved	Output2 Monitor	Output1 Monitor	Reserved	Reserved	Output2 Status	Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2F2 <sub>hex</sub>	0	Output4 Monitor	Output3 Monitor	Output2 Monitor	Output1 Monitor	Output4 Status	Output3 Status	Output2 Status	Output1 Status

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2F3 <sub>hex</sub>	0	Reserved	Reserved	Output6 Status	Output5 Status	Output4 Status	Output3 Status	Output2 Status	Output1 Status
	1	Reserved	Reserved	Output6 Monitor	Output5 Monitor	Output4 Monitor	Output3 Monitor	Output2 Monitor	Output1 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2F4 <sub>hex</sub>	0	Output8 Status	Output7 Status	Output6 Status	Output5 Status	Output4 Status	Output3 Status	Output2 Status	Output1 Status
	1	Output8 Monitor	Output7 Monitor	Output6 Monitor	Output5 Monitor	Output4 Monitor	Output3 Monitor	Output2 Monitor	Output1 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2F5 <sub>hex</sub>	0	Output8 Status	Output7 Status	Output6 Status	Output5 Status	Output4 Status	Output3 Status	Output2 Status	Output1 Status
	1	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Output10 Status	Output9 Status
	2	Output8 Monitor	Output7 Monitor	Output6 Monitor	Output5 Monitor	Output4 Monitor	Output3 Monitor	Output2 Monitor	Output1 Monitor
	3	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Output10 Monitor	Output9 Monitor



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2F6 <sub>hex</sub>	0	Output8 Status	Output7 Status	Output6 Status	Output5 Status	Output4 Status	Output3 Status	Output2 Status	Output1 Status
	1	Reserved	Reserved	Reserved	Reserved	Output12 Status	Output11 Status	Output10 Status	Output9 Status
	2	Output8 Monitor	Output7 Monitor	Output6 Monitor	Output5 Monitor	Output4 Monitor	Output3 Monitor	Output2 Monitor	Output1 Monitor
	3	Reserved	Reserved	Reserved	Reserved	Output12 Monitor	Output11 Monitor	Output10 Monitor	Output9 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2F7 <sub>hex</sub>	0	Output8 Status	Output7 Status	Output6 Status	Output5 Status	Output4 Status	Output3 Status	Output2 Status	Output1 Status
	1	Reserved	Reserved	Output14 Status	Output13 Status	Output12 Status	Output11 Status	Output10 Status	Output9 Status
	2	Output8 Monitor	Output7 Monitor	Output6 Monitor	Output5 Monitor	Output4 Monitor	Output3 Monitor	Output2 Monitor	Output1 Monitor
	3	Reserved	Reserved	Output14 Monitor	Output13 Monitor	Output12 Monitor	Output11 Monitor	Output10 Monitor	Output9 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2F8 <sub>hex</sub>	0	Output8 Status	Output7 Status	Output6 Status	Output5 Status	Output4 Status	Output3 Status	Output2 Status	Output1 Status
	1	Output16 Status	Output15 Status	Output14 Status	Output13 Status	Output12 Status	Output11 Status	Output10 Status	Output9 Status
	2	Output8 Monitor	Output7 Monitor	Output6 Monitor	Output5 Monitor	Output4 Monitor	Output3 Monitor	Output2 Monitor	Output1 Monitor
	3	Output16 Monitor	Output15 Monitor	Output14 Monitor	Output13 Monitor	Output12 Monitor	Output11 Monitor	Output10 Monitor	Output9 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2F9 <sub>hex</sub>	0	Output8 Status	Output7 Status	Output6 Status	Output5 Status	Output4 Status	Output3 Status	Output2 Status	Output1 Status
	1	Output16 Status	Output15 Status	Output14 Status	Output13 Status	Output12 Status	Output11 Status	Output10 Status	Output9 Status
	2	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Output18 Status	Output17 Status
	3	Output8 Monitor	Output7 Monitor	Output6 Monitor	Output5 Monitor	Output4 Monitor	Output3 Monitor	Output2 Monitor	Output1 Monitor
	4	Output16 Monitor	Output15 Monitor	Output14 Monitor	Output13 Monitor	Output12 Monitor	Output11 Monitor	Output10 Monitor	Output9 Monitor
	5	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Output18 Monitor	Output17 Monitor



**Safety Analog I/O Device, Type: 2A<sub>Hex</sub>**

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2FA <sub>hex</sub>	0	Output8 Status	Output7 Status	Output6 Status	Output5 Status	Output4 Status	Output3 Status	Output2 Status	Output1 Status
	1	Output16 Status	Output15 Status	Output14 Status	Output13 Status	Output12 Status	Output11 Status	Output10 Status	Output9 Status
	2	Reserved	Reserved	Reserved	Reserved	Output20 Status	Output19 Status	Output18 Status	Output17 Status
	3	Output8 Monitor	Output7 Monitor	Output6 Monitor	Output5 Monitor	Output4 Monitor	Output3 Monitor	Output2 Monitor	Output1 Monitor
	4	Output16 Monitor	Output15 Monitor	Output14 Monitor	Output13 Monitor	Output12 Monitor	Output11 Monitor	Output10 Monitor	Output9 Monitor
	5	Reserved	Reserved	Reserved	Reserved	Output20 Monitor	Output19 Monitor	Output18 Monitor	Output17 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2FB <sub>hex</sub>	0	Output8 Status	Output7 Status	Output6 Status	Output5 Status	Output4 Status	Output3 Status	Output2 Status	Output1 Status
	1	Output16 Status	Output15 Status	Output14 Status	Output13 Status	Output12 Status	Output11 Status	Output10 Status	Output9 Status
	2	Output24 Status	Output23 Status	Output22 Status	Output21 Status	Output20 Status	Output19 Status	Output18 Status	Output17 Status
	3	Output8 Monitor	Output7 Monitor	Output6 Monitor	Output5 Monitor	Output4 Monitor	Output3 Monitor	Output2 Monitor	Output1 Monitor
	4	Output16 Monitor	Output15 Monitor	Output14 Monitor	Output13 Monitor	Output12 Monitor	Output11 Monitor	Output10 Monitor	Output9 Monitor
	5	Output24 Monitor	Output23 Monitor	Output22 Monitor	Output21 Monitor	Output20 Monitor	Output19 Monitor	Output18 Monitor	Output17 Monitor

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2FC <sub>hex</sub>	0	Output8 Status	Output7 Status	Output6 Status	Output5 Status	Output4 Status	Output3 Status	Output2 Status	Output1 Status
	1	Output16 Status	Output15 Status	Output14 Status	Output13 Status	Output12 Status	Output11 Status	Output10 Status	Output9 Status
	2	Output24 Status	Output23 Status	Output22 Status	Output21 Status	Output20 Status	Output19 Status	Output18 Status	Output17 Status
	3	Reserved	Reserved	Reserved	Reserved	Output28 Status	Output27 Status	Output26 Status	Output25 Status
	4	Output8 Monitor	Output7 Monitor	Output6 Monitor	Output5 Monitor	Output4 Monitor	Output3 Monitor	Output2 Monitor	Output1 Monitor
	5	Output16 Monitor	Output15 Monitor	Output14 Monitor	Output13 Monitor	Output12 Monitor	Output11 Monitor	Output10 Monitor	Output9 Monitor
	6	Output24 Monitor	Output23 Monitor	Output22 Monitor	Output21 Monitor	Output20 Monitor	Output19 Monitor	Output18 Monitor	Output17 Monitor
	7	Reserved	Reserved	Reserved	Reserved	Output28 Monitor	Output27 Monitor	Output26 Monitor	Output25 Monitor



Safety Analog I/O Device, Type: 2A<sub>Hex</sub>

Instance	Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2FD <sub>hex</sub>	0	Output8 Status	Output7 Status	Output6 Status	Output5 Status	Output4 Status	Output3 Status	Output2 Status	Output1 Status
	1	Output16 Status	Output15 Status	Output14 Status	Output13 Status	Output12 Status	Output11 Status	Output10 Status	Output9 Status
	2	Output24 Status	Output23 Status	Output22 Status	Output21 Status	Output20 Status	Output19 Status	Output18 Status	Output17 Status
	3	Output32 Status	Output31 Status	Output30 Status	Output29 Status	Output28 Status	Output27 Status	Output26 Status	Output25 Status
	4	Output8 Monitor	Output7 Monitor	Output6 Monitor	Output5 Monitor	Output4 Monitor	Output3 Monitor	Output2 Monitor	Output1 Monitor
	5	Output16 Monitor	Output15 Monitor	Output14 Monitor	Output13 Monitor	Output12 Monitor	Output11 Monitor	Output10 Monitor	Output9 Monitor
	6	Output24 Monitor	Output23 Monitor	Output22 Monitor	Output21 Monitor	Output20 Monitor	Output19 Monitor	Output18 Monitor	Output17 Monitor
	7	Output32 Monitor	Output31 Monitor	Output30 Monitor	Output29 Monitor	Output28 Monitor	Output27 Monitor	Output26 Monitor	Output25 Monitor

### 6-7.8 Mapping I/O Assembly Data Attribute Components

The following table indicates the I/O assembly Data attribute mapping for the Safety Analog I/O device for the input assemblies with safety input data and no safety status.

**Table 6-7.28 Instance Numbers 180 -18D<sub>HEX</sub>, 1C0 -1CD<sub>HEX</sub>, 200 -20D<sub>HEX</sub>, 240 -24D<sub>HEX</sub>, 280 -28D<sub>HEX</sub>**

Data Component	Class		Instance	Attribute	
Name	Name	Number	Number	Name	Number
Safety Input <i>N</i>	Safety Analog Input Point	49hex	<i>N</i>	Safety Analog Input Value	2

The following table indicates the I/O assembly Data attribute mapping for the Safety Analog I/O device for the input assemblies with safety input data and safety input status.

**Table 6-7.29 Instance Numbers 190 -19D<sub>HEX</sub>, 1D0 -1DD<sub>HEX</sub>, 210 -21D<sub>HEX</sub>, 250 -25D<sub>HEX</sub>, 290 -29D<sub>HEX</sub>**

Data Component	Class		Instance	Attribute	
Name	Name	Number	Number	Name	Number
Safety Input <i>N</i>	Safety Analog Input Point	49hex	<i>N</i>	Safety Analog Input Value	2
Safety Input Status	Safety Analog Input	49hex	<i>N</i>	Input Point Status	5

The following table indicates the I/O assembly Data attribute mapping for the Safety Analog I/O device for the output assemblies with safety output data.

**Table 6-7.30 Instance Numbers 1B0 -1BD<sub>HEX</sub>, 1F0 -1FD<sub>HEX</sub>, 230 -23D<sub>HEX</sub>, 270 -27D<sub>HEX</sub>, 2B0 -2BD<sub>HEX</sub>**

Data Component	Class		Instance	Attribute	
Name	Name	Number	Number	Name	Number
Safety Output <i>N</i>	Safety Analog Output Point	TBDhex	<i>N</i>	Safety Output Value	TBD



This page is intentionally left blank



## **Volume 5: CIP Safety**

# **Chapter 7: Safety Device Configuration & Electronic Data Sheets**

---



## Contents

7-1	Safety Configuration Process.....	3
7-1.1	Introduction to Safety Configuration .....	3
7-1.1.1	Configuration Goals .....	3
7-1.1.2	Configuration Overview .....	3
7-1.1.3	User Configuration Guidelines .....	5
7-1.1.4	Configuration Process SIL3 Justification .....	6
7-1.2	Device Functions for Tool Configuration .....	7
7-1.2.1	Password Security .....	7
7-1.2.2	SNCT Interface Services .....	7
7-1.2.3	Configuration Lock .....	8
7-1.2.4	Configuration Ownership .....	10
7-1.2.5	Configuration Mode .....	10
7-1.3	Measures Used To Ensure Integrity of Configuration Process .....	10
7-1.3.1	Safety Configuration Identifier (SCID) .....	10
7-1.3.2	Safety Configuration CRC (SCCRC) .....	11
7-1.3.3	Safety Configuration Timestamp (SCTS).....	11
7-1.3.4	System-Wide Unique "Safety Network Number" (SNN).....	11
7-1.3.5	System-Wide "Unique Node Identifier" (UNID).....	11
7-1.3.6	Connection Parameters CRC (CPCRC).....	12
7-1.4	Download Process.....	13
7-1.4.1	SNCT download to originators and targets.....	13
7-1.4.2	SNCT Downloads to Originators which do Type 1 Target Configuration .....	14
7-1.5	Verification Process .....	16
7-1.5.1	User Configuration Verification and Alternatives .....	17
7-1.5.2	Verification Process.....	19
7-1.6	Configuration Error Analysis.....	21
7-1.6.1	Errors .....	21
7-1.6.2	Detection Measures .....	22
7-1.7	Diagnostic Software Protections (Not Safety Related) .....	24
7-1.8	Device Memory Architecture Considerations .....	24
7-2	Electronic Data Sheets .....	26
7-2.1	General Rules for EDS based Safety devices.....	26
7-2.1.1	Safety Configuration Assembly Definition .....	26
7-2.1.2	Configuration CRC.....	26
7-2.1.3	Password Encryption for EDS Devices .....	26
7-2.2	General EDS Extensions for Safety .....	26
7-2.2.1	Extension to [File] Section for Safety.....	26
7-2.2.2	Extensions to [Device Classification] Section for Safety .....	27
7-2.2.3	Extension to [ParamClass] Section for Safety .....	28
7-2.2.4	Extension to [Connection Manager] Section for Safety .....	28



## 7-1 Safety Configuration Process

### 7-1.1 Introduction to Safety Configuration

The purpose of this section is to identify the TUV certified Safety Network Configuration process and the configuration data flows through the safety system. This section defines the system and the software requirements of the Safety Network Configuration interface and how data is protected to ensure the safety system meets SIL3.

The configuration process is comprised of two parts: 1) download and 2) testing and verification.

#### Download Process:

During the download portion of the configuration process the configuration data is transferred from one of the following two sources:

1. The Safety Network Configuration Tool (SNCT) to the device being configured
2. From the originator to the target device.

#### Verification Process:

During the verification stage the user will functionally test the device in the system, and visually verify that the correct configuration data has been transferred to the device being configured.

Together the protections in these two parts ensure that devices are suitably configured for use in a SIL3 environment.

#### 7-1.1.1 Configuration Goals

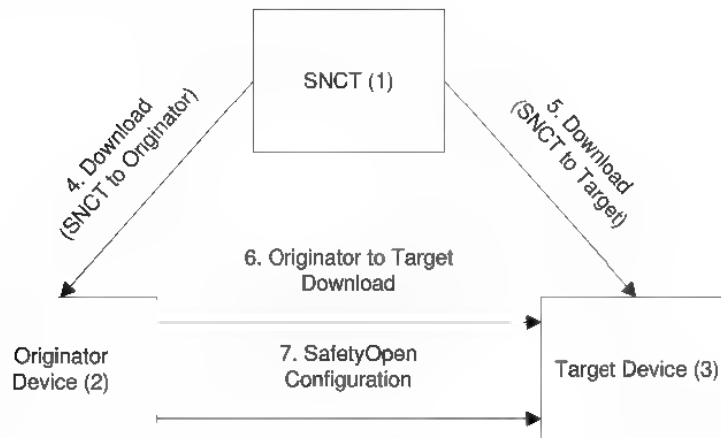
Goal	Description
Goal 1	To ensure with high integrity that the intended configuration created by the user with the software was correctly sent to and saved in the safety device.
Goal 2	To have a common download process that is the same for the transfer of configuration data to all safety devices.
Goal 3	To have a common download process that is the same for the transfer of configuration data from a software tool or an originating device during configuration.
Goal 4	To ensure with SIL3 integrity that a target device has the intended configuration prior to opening a safety connection. The measures used during configuration download should be reused in the safety connection establishment process.
Goal 5	To ensure unique identification of configuration with or without data being included within a connection establishment

#### 7-1.1.2 Configuration Overview

Safety Configuration begins with the downloading of configuration data into a safety device. Figure 7-1.1 shows the possible configuration data transfers into safety devices. *The numbers in the figure are references to the discussion numbering below.*



Figure 7-1.1 Configuration Data Transfers



The devices involved in the configuration process are shown in Figure 7-1.1.

- The SNCT (1) Safety Network Configuration Tool and the process described in this specification allows this tool to be non-safety critical.
- The Originator (2) is a device that is capable of configuring Target devices (3) while initiating connections to them. Because these devices can hold configuration data for Targets (for targets that are not configured by the SNCT directly), they have additional procedures to complete the configuration process.
- The Target device (3) is one that can be either configured directly by the SNCT or can be configured by an Originator (2) owner. The ownership concept plays an important role in assuring the integrity of “originator configured” targets.

There are two basic configuration procedures (refer to Figure 7-1.1)

- SNCT-to-Originator (flow 4) and SNCT-to-Target (flow 5). This procedure is one that configures a single device and these devices hold no configuration data for other devices.
- SNCT-to-Originator-to-Target (flow 4 plus flow 6 or 7). This special procedure is for configuring an originator device that holds configuration data for a target.
- Flow 6 is a procedure that duplicates the sequence of operations that would otherwise be done in flow 5.
- Flow 7 is a procedure by which the configuration is performed as part of the processing of a SafetyOpen.
- There are two ways that a replacement device can be reconfigured from the originator
- Flow 6 is a procedure that duplicates the sequence of operations that would otherwise be done in flow 5.
- Flow 7 is a procedure by which the configuration is performed as part of the processing of a SafetyOpen.



### 7-1.1.3 User Configuration Guidelines

This section describes user configuration guidelines that assure a CIP safety system is properly configured and protected. These guidelines are used to define the safety manual requirements.

1. The CIP safety system user must assure that SNCT-configured-devices (2) are locked after verification (related requirements SRS50 & SRS51).

SRS202 Devices that can be configured via the SNCT interface shall include a safety manual advisory that instructs the user to lock the device after verification has been completed.

2. The CIP safety system user must assure that all originator-configured-devices (3) have ownership (7) assigned to the intended originator. (related requirements SRS92 & SRS44)

SRS203 Devices that can be configured by a Type 1 SafetyOpen shall include a safety manual advisory that instructs the user to verify that all originator-configured safety devices have their ownership assignments as part of the final verification process.

3. The user must visually confirm (4) the sent configuration data and confirm it was downloaded correctly (related requirements SRS38).

SRS204 All CIP safety devices shall include a safety manual advisory that instructs the user to visually verify that all configuration data was downloaded correctly.

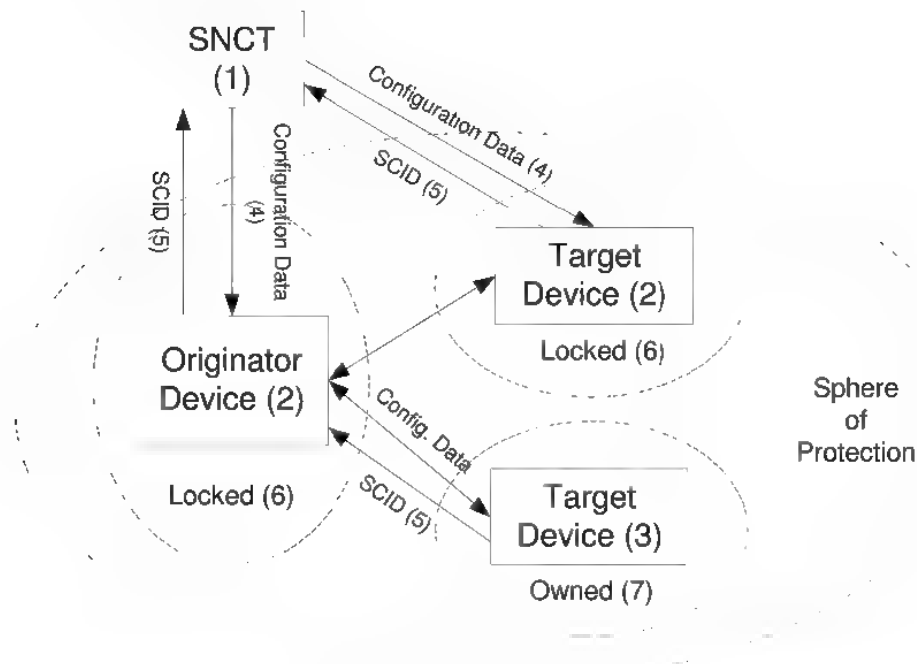
4. The user must functionally test their application (related requirements SRS42 & SRS43).

Refer to section 7-1.5, Verification Process for related safety manual requirements.

5. The user must assure the SCID (5) check by target and originator is done (related requirements SRS44).

Refer to section 7-1.5, Verification Process for related safety manual requirements.

Figure 7-1.2 Protection Measures in Safety Devices





#### **7-1.1.4 Configuration Process SIL3 Justification**

The CIP safety device being downloaded has safety measures defined which ensures integrity of data and data transfers as specified by IEC61508 SIL3.

The SNCT on the other hand, will not be certified due to the limitations of personal computer hardware, the operating system utilized, and the development tools. Because of these factors, the techniques employed to configure a safety device require that the safety device and user procedures, not the workstation software, enforce the protection measures.

SIL3 integrity is achieved by meeting the following requirements:

- SRS36 A receiving SIL3 device shall calculate a SCCRC with SIL3 integrity and compare it to the SCCRC within the SCID .
- SRS37 For Type 1 configured or tool configured devices, configuration validation of the downloaded safety-relevant data from the software (or originator) shall be performed with SIL3 integrity .
- SRS38 When a SIL3 device is configured directly from a workstation, the device safety manual shall instruct the user to compare the transferred SCID and configuration data with the SCID and configuration data originally viewed in the workstation.
- SRS39 The SNCT software shall provide a facility to allow the user to confirm downloaded configurations are correct.
- SRS40 SIL3 devices shall store/restore its configuration data with SIL3 integrity .
- SRS41 SIL3 devices shall handle all device state transitions with SIL3 integrity

Sub-systems that are required to be examined and certified to SIL3

- Target device configuration processing
- Target device configuration storage
- Target device configuration validation
- Target device modes

Sub-systems that are not required to be certified SIL3

- **Explicit messaging**

**Justification:** The explicit messaging subsystem is simply a means of transferring the data, all integrity measures are performed by other system components and any errors such as corruption or loss of messages will be detected by the safety measures.

- **Software User Interface**

**Justification:** The software interface is a means to display and enter data that will later be confirmed by other means. In itself, it does not require SIL3 integrity, but it will be used in conjunction with a diverse read back and display mechanism to form a high integrity function.

- **Software Tool configuration storage**

**Justification:** Any configuration data is stored with a safety related CRC that is verified to match the safety related CRC in the device. Any corruption of the data will be detected by the device's SIL3 CRC checking process.



## **7-1.2 Device Functions for Tool Configuration**

The process by which the SNCT interacts with originators and targets are controlled by facilities defined in the Safety Supervisor Object. Features such as Password protection and the Configuration Lock attribute combine to provide security for this interface. Since the Safety Supervisor is provided in the baseline profile (refer to Chapters 5 & 6) for both safety targets and originators, the interface details are common. This section will define these features and how they are used by the SNCT.

### **7-1.2.1 Password Security**

The SNCT interface defines the following functions to **optionally** provide password security:

- Password functionality added
  - SRS17 Support for one password shall be implemented by devices supporting the SNCT interface
  - Passwords submitted to devices supporting the SNCT interface shall be values obtained from running the encryption algorithm defined in Section 7-2.1.3 and Appendix E.
  - SRS20 The default password value in devices supporting the SNCT interface shall be zero
    - SRS21 Software shall transparently insert “zero” in all services which require one until a user sets it to something different
- SRS22 The SNCT interface shall support a Password service to set passwords which takes 2-password parameters; Old PW and New PW
- SRS23 The SNCT interface shall provide a Reset Password service which takes a vendor specific field to reset the password
  - Each vendor shall choose a method to reset passwords and instruct the user what to fill in to cause the device to reset the password. The software shall only insert whatever value the user enters into the reset field (refer to the Reset\_Password service in the Safety Supervisor object definition in Chapter 5).
  - The safety supervisor commands that have a password parameter define the processing requirements and error responses for these parameters.

### **7-1.2.2 SNCT Interface Services**

- SRS24 The SNCT interface shall provide a Configure Request service which requires password, TUNID, and OUNID to execute
- SRS130 In the SNCT interface, an unowned device shall capture the OUNID as the device owner when the first Configure\_Request is received.
- SRS26 The SNCT interface shall provide a Validate Configuration service which requires the SCID, SCCRC, and the SCTS to execute .
- SRS131 The SNCT interface shall provide an Apply service that causes the device to save the configuration to NV memory.
- SRS27 The SNCT interface shall provide a special Reset service which requires the password and TUNID to execute and supports the 2 common Reset types along with one that can reset to the default but preserve the password .
- SRS28 The SNCT interface defines an optional a Mode Change service which, if implemented, shall require a password to execute .



### **7-1.2.3 Configuration Lock**

The SNCT interface defines the following functions for Configuration Lock. Its intent is to provide a way for the user to flag that a tool-generated configuration has been tested and verified as valid. From that point, the signature (SCID) reflects a verified configuration.

- SRS25 The SNCT interface shall provide a Configuration Lock service which requires the password and TUNID to execute
- SRS132 The SNCT Configuration Lock Attribute shall have a default value of “Unlocked”
- SRS31 Safety devices which support the SNCT interface shall require that the “Configuration Lock” attribute be cleared before any command to change a safety device to the Configure State (Configure\_Request) will be accepted.

The Configuration Lock has two purposes

- Serves to indicate that the user has tested and confirmed the Tool-based configuration in this device is correct.
- “Tool Only” mode requirement. SRS32 When the Configuration Lock attribute is set, the device shall reject any Type 1 Safety Open messages.

These two features make it clear that for the SNCT to configure any safety node, it shall support features to set and clear the Configuration Lock attribute, and shall present the proper password to do so. These features are illustrated in the sequence diagram shown in Figure 7-1.5.

#### **7-1.2.3.1 Effect of Configuration Lock on Device Behavior**

Figure 7-1.3 shows the effect that the “configuration locked” condition has on the ability of a device to be configured; there are requirements defined in the Safety Supervisor object that cause safety devices to reject all attempts to change a locked configuration.



Figure 7-1.3 Configuration, Testing and Locked Relationships.

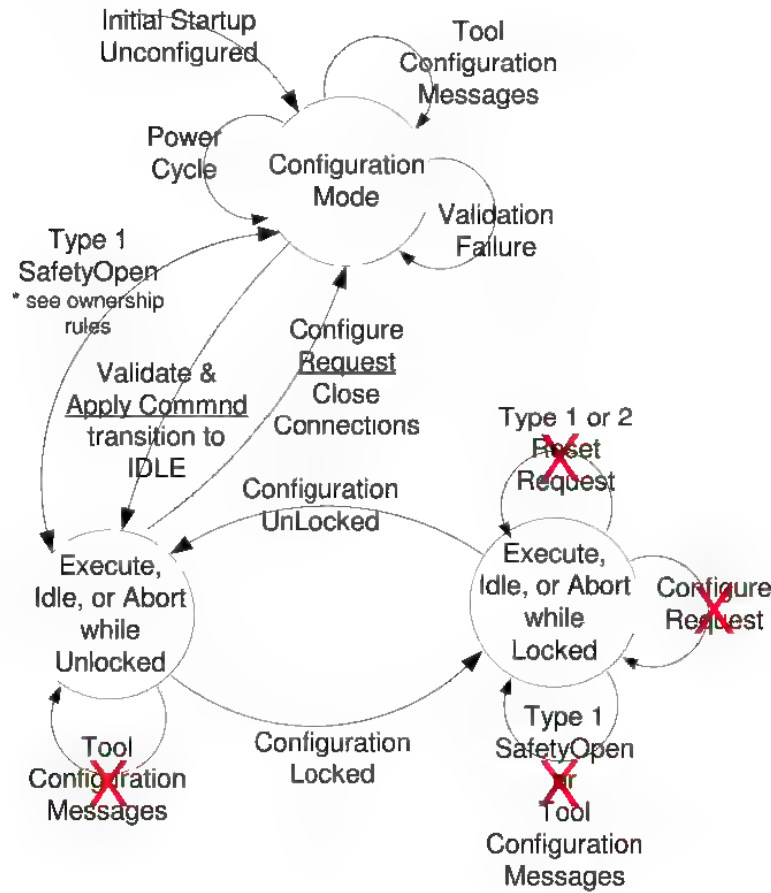


Table 7-1.1 shows the effect that ownership has on a device's ability to be configured. It shows that owned devices can only be Type 1 configured by the originator that has the owner UNID. A locked device can never be configured regardless.

Table 7-1.1 Configuration Owner Control vs. Device State

Ownership Rules	Configure mode	Execute/Idle Mode Unlocked	Execute/Idle Mode Locked
<b>Configuration Un-owned</b>	Tool configuration or Type 1 acceptance by anyone	Tool configuration or Type 1 acceptance by anyone	No Tool configuration, No Type 1 accepted
<b>Originator Owned</b>	Tool configuration or Type1 by owner only	No Tool configuration (must reset first) or Type1 by owner only	No Tool configuration, No Type 1 accepted
<b>Software Tool Owned</b>	Tool configuration only No Type 1 accepted	Tool configuration only No Type 1 accepted	No Tool configuration, No Type 1 accepted



#### **7-1.2.4 Configuration Ownership**

The SNCT interface defines functionality for handling the configuration owner identifier for a device. This function has been defined to allow a device to provide exclusive configuration rights to a particular configuration source. The source could also be assigned exclusively to a configuration tool. The following functions are provided:

- Configuration UNID attribute
  - The default is to allow any device to be an owner (refer to SRS70 & SRS134)
    - Device are required to capture the first configuration source as the owner (refer to SRS205)
  - Attributes and services in the Safety Supervisor are defined which allow Tools to be the exclusive configuration source
- Even though an owner has been assigned as a configuration source, the SNCT tool has the ability to override a configuration. The SNCT shall provide protection measures to assure the user cannot perform an override without warnings and the proper password.

#### **7-1.2.5 Configuration Mode**

SRS136 All safety devices shall support a non-volatile configuration mode, this means that once a device is commanded to enter the configuration mode it shall remain in the configuration mode until the configuration is successfully validated.

### **7-1.3 Measures Used To Ensure Integrity of Configuration Process**

This section describes the various protections that ensure the integrity of the download process. This section also identifies the portions of the system supporting the download process that are required to be certified to a SIL3 level.

#### **7-1.3.1 Safety Configuration Identifier (SCID)**

In safety systems it is necessary to clearly identify the configuration. The identification is used during several operations:

Download from tools – This gives the user the ability to check that the tool and device agree on the information downloaded.

Device Replacement – If the tool is used for device replacement, the user has the ability to verify that the configuration is the correct one to download. If the originator is used to automatically reconfigure a device, the SCID is used to indicate that a reconfiguration is necessary and ensure the integrity of the operation

Connection Establishment – The originator and target use the SCID to ensure that both devices are using the same configuration.

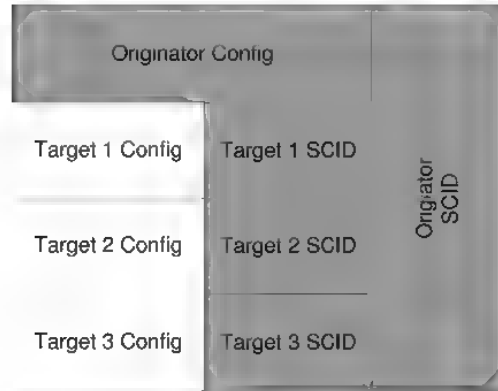
The SCID is a concatenation of the SCTS and the SCCRC described in Section 7-1.3.3 and Section 7-1.3.2 respectively.



#### 7-1.3.1.1 Originator and Target SCID Coverage

When originators are used to configure targets, Figure 7-1.4 illustrates how the SCIDs provide coverage of the various configurations. The SCID for targets that are configured by originators are not calculated in the originators, the calculation is performed in the workstation. The target configurations are protected by their respective SCIDs.

Figure 7-1.4 Originator's Configuration Data



#### 7-1.3.2 Safety Configuration CRC (SCCRC)

This is a CRC that covers the device configuration data that is contained in a SafetyOpen. It is part of the SCID for the device. The SCCRC is described in Chapter 2, Section 2-3.1.1.

**Safety Function** – The SCCRC is used to ensure the correctness of the safety application data.

#### 7-1.3.3 Safety Configuration Timestamp (SCTS)

The Safety Configuration Time Stamp is a safety related signature that identifies the revision of the device configuration contained in either a Safety Open or in a safety device. It is part of the SCID for the device configuration. Refer to FRS162 and Chapter 2 Section 2-3.1.1

**Safety Function** – The SCTS is used to ensure the uniqueness of the safety application data.

#### 7-1.3.4 System-Wide Unique "Safety Network Number" (SNN)

The Safety Network Number uniquely identifies a network across all networks in the safety system. The end user is responsible for assigning a unique number to each safety network or safety sub-net within a system. Refer to FRS162 and Chapter 2 Section 2-3.1.1

**Safety Function** – The UNID is used to uniquely identify originators and/or targets during the connection establishment process

#### 7-1.3.5 System-Wide "Unique Node Identifier" (UNID)

Uniquely identifies a node across all networks in the safety system. This value is made up of the Safety Network Number (SNN) and the Node Address of the device. This value is a structure consisting of the 6-octet SNN and the 4-octet Node Address. The 4-octet node address is sized to accommodate all forms of CIP network addresses.



```
struct UNID
{
    S SNN      SNN;
    UDINT      MACID;
};
```

MACIDs smaller than 4-octets (i.e. DeviceNet) shall be aligned to the least significant octet in little endian order. The unused octets shall be zero filled.

**Safety Function** – The UNID is used to uniquely identify originators and/or targets during the connection establishment process

Other uses of UNID:

**TUNID** = Target's Unique Network Identifier

**Safety Function** – Identifies the target in the SafetyOpen & Configuration mode message

**OUNID** = Originator's Unique Network Identifier

**Safety Function** – Identifies the originator in the SafetyOpen.

**OCPUNID** = Output Connection Point Owning UNID

**Safety Function** – Identifies the owner of the output connection in the target. Set to the OUNID of the originator in the SafetyOpen that obtains ownership of the output connection point. Used to prevent an unauthorized originator to obtain ownership of an output connection point

**CFUNID** = Configuration Owning UNID

**Safety Function** – Identifies the owner of the safety configuration in the target, this should be equal to the OUNID in normal operation. This is set to the OUNID of the originator in the SafetyOpen that includes configuration data and is used to prevent an unauthorized originator from modifying the configuration.

#### **7-1.3.6 Connection Parameters CRC (CPCRC)**

The Connection Parameter CRC is a safety related CRC covering the CIP connection establishment data in the Safety Open. This CRC will also cover the additional parameters required by the safety protocol. The CRC is defined in Appendix E

**Safety Function** – The CPCRC is used to ensure the integrity of the connection data contained within the Safety Open.



## 7-1.4 Download Process

The safety download process ensures that the configuration data is transferred with high integrity from the SNCT or the Originator to the device being configured. At the successful completion of the download process the configuration is considered valid. There are two download processes used with safety devices:

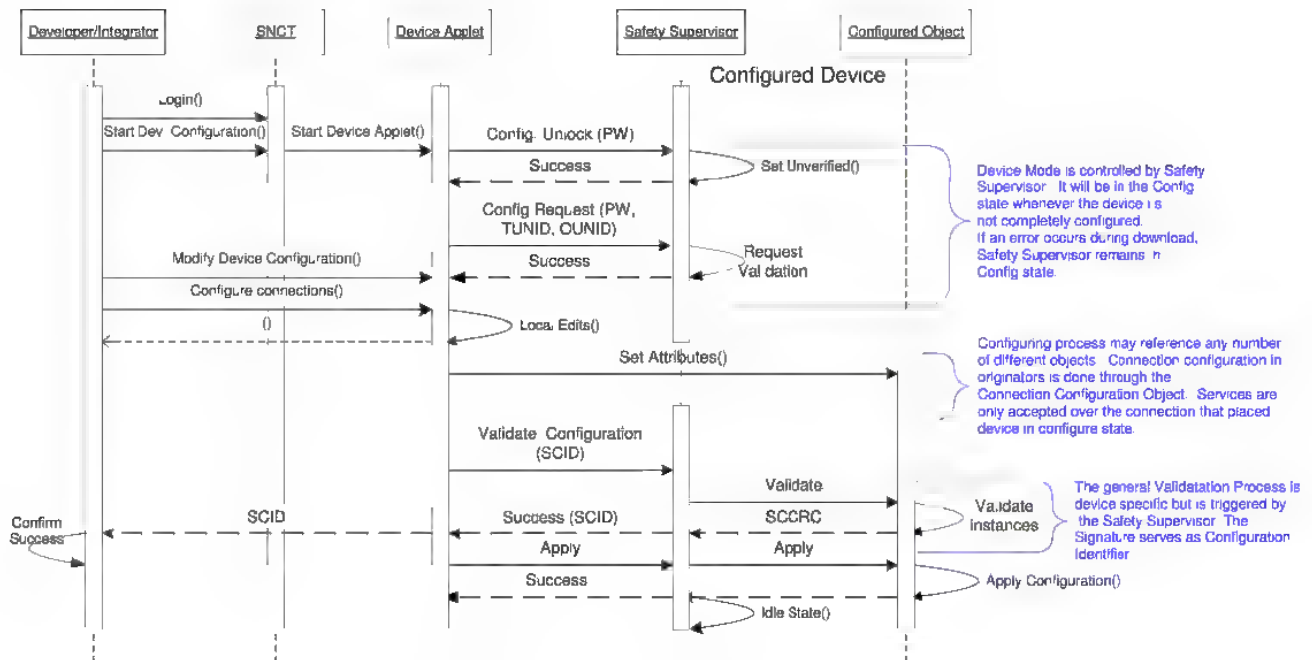
- SNCT downloads to originators and targets
- Originator downloads to targets using a Type 1 SafetyOpen

The following sections will describe the details of the processes that are used for the download of safety devices.

### 7-1.4.1 SNCT download to originators and targets

Figure 7-1.5 show the process of downloading a safety device from the SNCT. The download process between the SNCT and the Configured Device assures that the data is instantiated in the Configured Device's memory with high integrity.

**Figure 7-1.5 SNCT to Device Download Process**





#### **7-1.4.1.1 SNCT to Device Download Process Steps**

1. Clear Configuration Lock Attribute
  - a. Device: Configuration Lock Attribute cleared.
2. Enter configuration state with password, TUNID, OUNID
  - a. Device: No safety I/O connections can be active.
  - b. Device: Device saves Config mode to NV storage
    - Until “Validate” & “Apply”, power cycles go back to config mode
3. Transfer of configuration data
  - a. Tool: Calculate “proposed SCCRC” for data (SCCRC part of SCID)
  - b. Device: SRS138 Any explicit messages are allowed to set configuration in device, but device shall only accept safety-related explicit messages over the connection that put it into the configuration mode.
4. Validate configuration data
  - a. Tool: Send Validate\_Request with SCID (which contains proposed SCCRC)
  - b. Device: Analyze configuration data
  - c. Device: Calculates SCCRC value and compares to “proposed SCCRC”
  - d. Device: send SCID in Validate\_Reply
  - e. Tool: Compares returned SCCRC to “proposed SCCRC”
5. Apply Configuration
  - a. Device: Save configuration to NV storage with SCID
  - b. Device: Mark configuration as valid.
  - c. Device: Set mode to Idle
  - d. Tool: Capture and store the SCID with the configuration file.

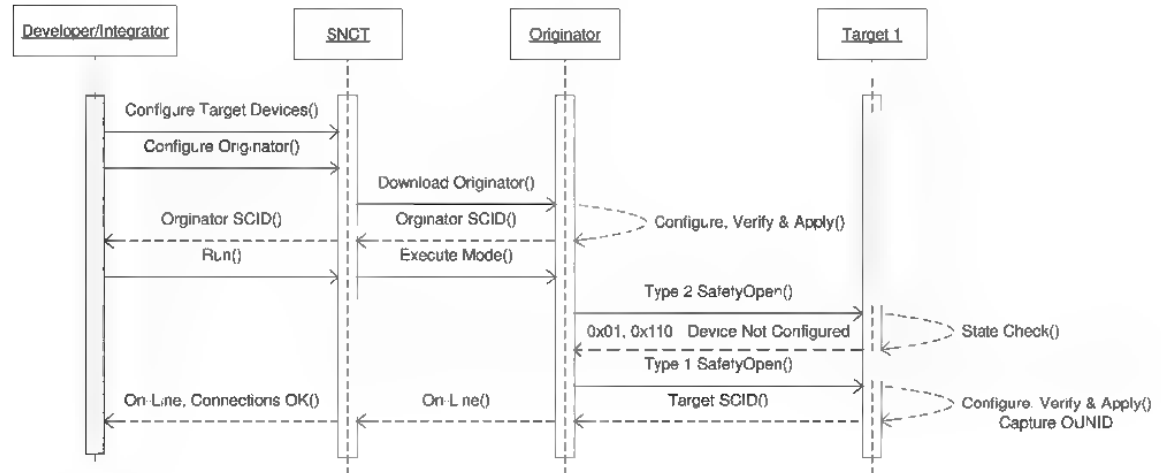
#### **7-1.4.2 SNCT Downloads to Originators which do Type 1 Target Configuration**

Figure 7-1.6 show the process by which Originators are initially configured and how the Type 1 Safety Open is used to download Target devices. Once the user finishes defining the Originator and Target configurations in the SNCT, it is sent to the Originator. At this point the originator SCID can be confirmed by the user. This process ensures that the configuration data in the SNCT and originator match with a high degree of integrity.

Then the user must place the Originator into execute mode to get the Target devices downloaded. If the connection is successful, the Target configuration/SCID was accepted by the target and confirmed by the originator. This action also established the Originator as the Target’s owner and prevents the Target from being downloaded by any other device except the SNCT.



Figure 7-1.6 SNCT Downloads to Originators that perform Type 1 Configuration



#### 7-1.4.2.1 SNCT Downloads to Originators that perform Type 1 Configuration

1. Clear Configuration Lock Attribute on originator
  - a. Device: Configuration Lock Attribute cleared.
2. Enter configuration state with password, TUNID, OUNID
  - a. Device: No safety I/O connections can be active.
  - b. Device: Device saves “Configuring” mode to NV storage
    - i. Until “Validate” & “Apply”, power cycles go back to “Configuring mode”
3. Transfer of configuration data
  - a. Tool: Calculate “proposed SCCRC” for data (SCCRC part of SCID)
  - b. Tool: Any explicit messages are allowed to set configuration in device, but device shall only accept safety-related explicit messages over the connection that put it into the configuration mode.
4. Validate configuration data
  - a. Tool: Send Validate \_Request with SCID (which contains proposed SCCRC)
  - b. Device: Analyze configuration data
  - c. Device: Calculates SCCRC value and compares to “proposed SCCRC”
  - d. Device: send SCID in Validate\_Reply
  - e. Tool: Compares returned SCCRC to “proposed SCCRC”
5. Apply Configuration
  - a. Device: Save configuration to NV storage with SCID
  - b. Device: Mark configuration as valid.
  - c. Device: Set mode to Idle
  - d. Tool: Capture and store the SCID with the configuration file.
6. Set Mode to Run
  - a. Originator: Sends Type 2 SafetyOpens (refer to Chapter 2, Section 2-6.2)
  - b. Target returns one of the following errors (assumes no others are found, refer to SRS11, SRS12, & SRS180 in Chapter 2):
    - Unconfigured: 0x01, 0x0110 – Device Not Configured
    - Prior configuration: 0x01, 0x80C - SCID Mismatch error
  - c. Originator: Sends Type 1 SafetyOpen



- d. Targets: Verify and Apply the configuration data
- e. Targets: Return target SCID# to originators
- f. Originator: Compare sent and received SCID#'s
- g. Originator: Return On-Line status to SNCT

### **7-1.5 Verification Process**

The verification process uses the data that was transferred during any of the download processes to allow the user to correctly verify that the downloaded data is correct, the system operation is correct and the system is ready for SIL 3 operation. There are three main parts to the configuration process:

- Configuration Data Verification
- Functional testing
- Configuration Locking

Key requirements for the safety manual:

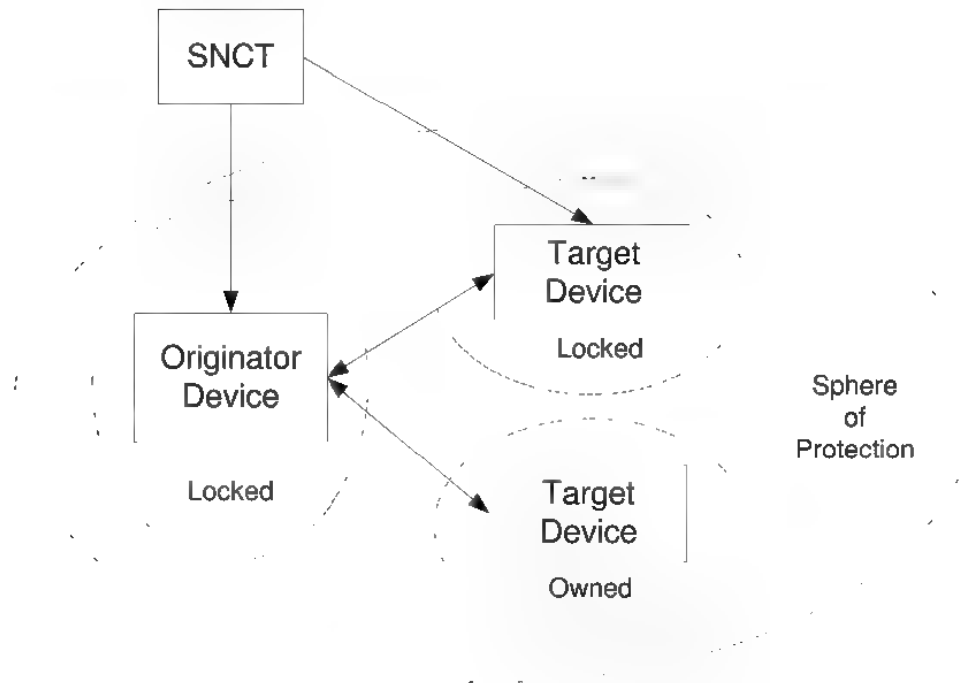
- SRS42 The device Safety Manual shall contain an advisory that user testing is the means by which all downloads are validated.
- SRS43 The device Safety Manual shall contain an advisory that the signature should only be considered “verified” (and configuration locked) after user testing

SRS44 The device User manual shall contain an advisory that configuring an originator with connection data and/or target configuration data must be downloaded to the target so it can be tested and verified. Only then can SCIDs from the target be confirmed.

As shown in Figure 7-1.7, when originators are tool-configured and targets are originator configured, only the originator is “locked” after verification and testing. The target in this case uses ownership assignments for protection. Also shown in Figure 7-1.7 is that Targets can also be tool-configured as well. In this case, these targets are also “locked” after verification and testing.



Figure 7-1.7 Protection from Locking & Ownership



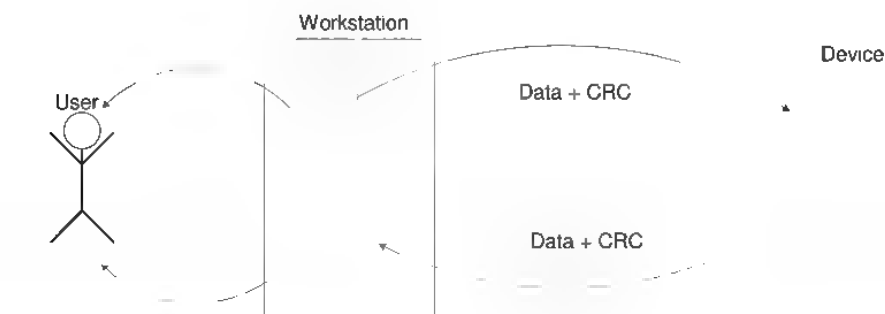
### 7-1.5.1 User Configuration Verification and Alternatives

The user needs to visually verify that the downloaded configuration is correct and is the intended configuration for the device (refer to Section 7-1.1.3). This section presents four alternatives that will fulfill the requirements for user verification of the device configuration data. In all cases the user needs to verify the functionality of the devices being configured through some kind of functional testing (refer to Section 7-1.1.3).

#### 7-1.5.1.1 Alternative 1 – Immediate Read Back and Diverse Comparison

The device configuration data is read back by the work station and presented to the user utilizing a diverse execution path in the workstation.

Figure 7-1.8 Example of read back and comparison of original and printout





1. One or a combination of the following options can be used to fulfill the user verification requirement:
2. User comparison of the original software display, and results of the read back that have been displayed by a diverse means.

Reading back the entire configuration, comparing it to what the software has and providing a printout of the device configuration for the user to compare with what was previously entered. The user must examine and compare the printout to the original data to satisfy the integrity requirements. Any automated comparison by the software tools is a diagnostic and shall not be relied upon in and of itself to ensure integrity.

The read back and display using a diverse means reduces the possibility of a systemic error in the workstation causing the same data corruption on data entry and user checking, which could cause an undetected error. If, for instance, a corruption occurred prior to the generation of the CRC the data would be interpreted as correct by the device and the same data and CRC would be sent back to the workstation. Automated checks of the original and read back data would not indicate an error, because the data matches the CRC. If the data was displayed using the same mechanism as the original display mechanism the error could still be masked and would be undetected. The diverse display paths should reduce this common cause of error.

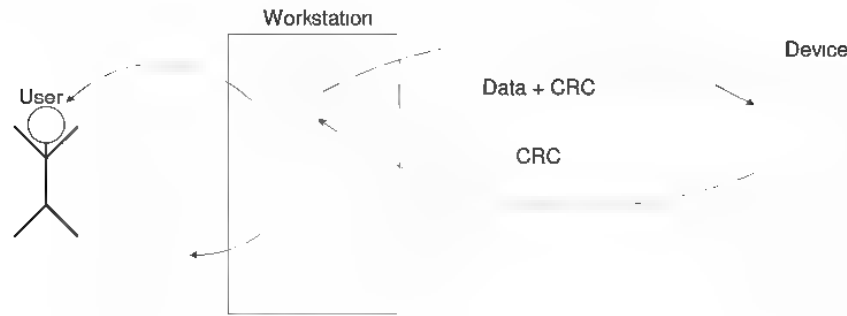
#### **7-1.5.1.2      Alternative 2 – Local Diverse Display**

It is possible to achieve the same level of integrity as defined in 7-1.5.1.1 since it can be shown that the configuration data on the workstation storage is identical to the configuration data in the device. The verification of configuration data is accomplished by writing the configuration data and CRC to the local drive in the workstation. The configuration data and CRC from the local drive are then sent to the device. The device then verifies the correct transfer by comparing the data and CRC. If the transfer to the device is correct, the device sends its calculated CRC back to the workstation to confirm that the data was sent correctly. At that point the workstation compares the device CRC to the stored CRC. At this time it can be assumed that the data in the workstation was correctly sent to the device. It is important to note that the CRC is only calculated once by the workstation and stored on the local drive. It is not recalculated for the purpose of transfers. This also ensures the integrity of the data on the local drive. At this point it can be assumed that the data in the workstation was correctly sent to the device.

For verification, the user shall view the “sent” configuration data using a diverse means to confirm that the correct configuration was sent to the device (refer to Section 7-1.1.3). Since the data on the workstation storage includes a CRC, a diverse means of display will allow the user to uncover any errors caused by the workstation when the data was entered into storage. This effectively duplicates the functionality of the full read back method without the overhead of doing a full data read back from the device to the workstation.



**Figure 7-1.9 Diverse display without full data read back**



### 7-1.5.1.3 Alternative 3 – Delayed Local Diverse Display

This alternative presents a method to use a combination of the TÜV preferred method and the Alternative 1 method described above. The goal of this method is to reduce the user interaction with the system.

Initially all of the devices in the system are unverified and are downloaded with the CRC checking as described in Alternative 1, the user verification of the uploaded data is not done at this time. Once the entire system is functionally tested by the user, the entire system will be verified by the software tool on the workstation. The verification step shall read back all of the configuration data from all of the devices, compare the read back data to the local drive copy, and present it to the user in a diverse manner for user examination. This read back of data shall be performed in the manner described in section 5.4 but shall be performed for all devices in the system at the same time.

This method has the advantage of allowing the user to complete the onerous visual comparison once, with the final tested system.

### 7-1.5.1.4 Alternative 4 – Delayed Read back

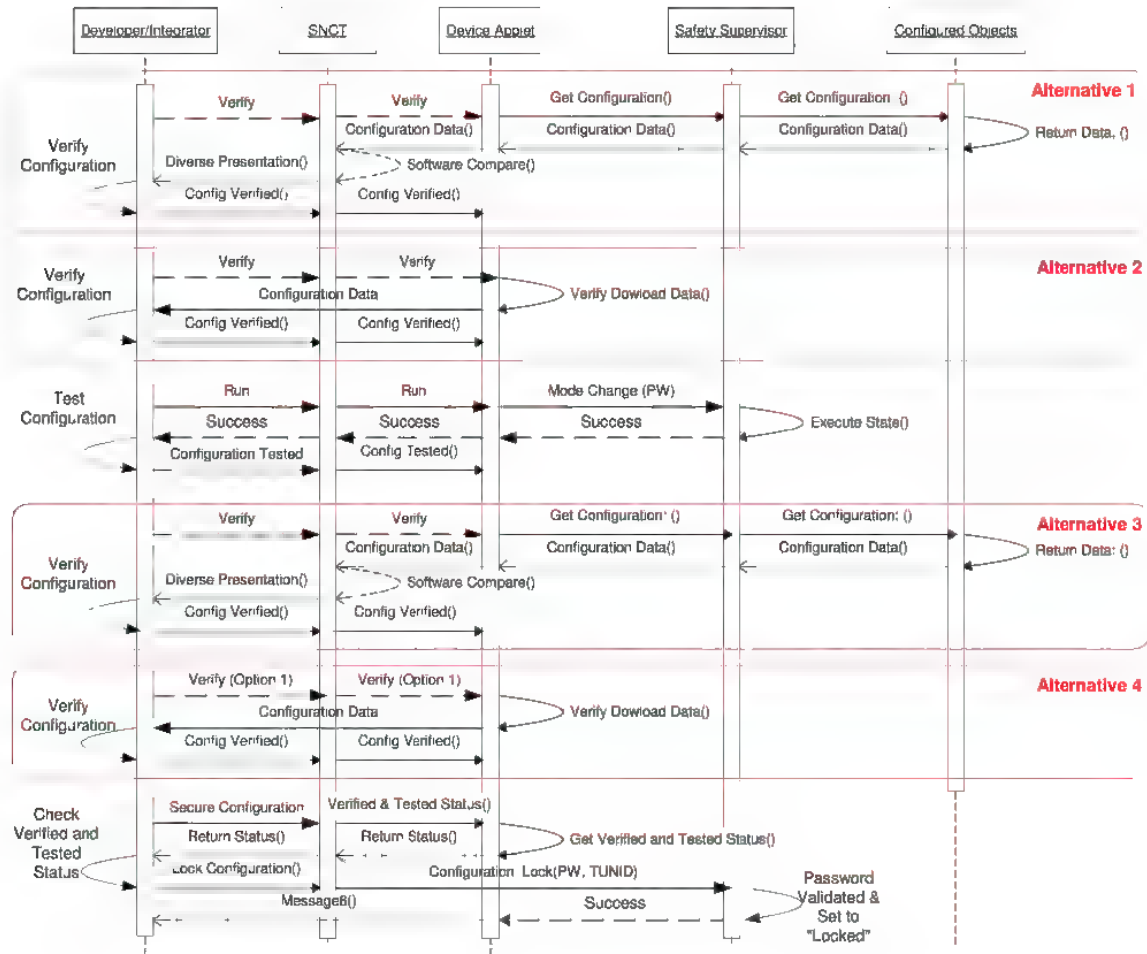
This alternative presents a method similar to the TÜV preferred method, but with the read back of the configuration data from the devices delayed until the testing of the device is preformed. Functionally this method is identical to the TÜV preferred method, but it allows the user to delay the visual verification until system testing is complete.

## 7-1.5.2 Verification Process

Figure 7-1.10 shows the complete process for verifying the configuration of safety devices. It is assumed that the data has been downloaded using the process described in section 7-1.4, ensuring that the SNCT and the device have the same configuration data. One alternative shall be selected for the user verification of the configuration data (refer to Section 7-1.1.3). Upon completion of this process the devices and system are ready for SIL 3 operation.



Figure 7-1.10 Verification Process including all alternatives





## 7-1.6 Configuration Error Analysis

The following table shows the errors and detection measures that are considered for the download and configuration process defined by the SNCT interface.

**Table 7-1.2 Errors and Detection Measures**

Errors	Detection Measures							
	User Authentication <sup>1</sup>	Identification of configuration	Identification of target	SCCRC protection on configuration (Part of SCID)	Configuration Locking or Device Ownership	Configuration session control	User Functional Testing	Read back of the device configuration
User misdirected configuration	X				X		X	
System misrouted configuration	X		X		X		X	X
Lost Configuration Message 0				X			X	X
Corrupted configuration				X			X	X
Configuration at inappropriate time						X		
User loads wrong configuration		X					X	X
Configuration process is interrupted						X		

<sup>1</sup> User Authentication is an optional feature and only applies if the feature is used by the end user

### 7-1.6.1 Errors

#### 7-1.6.1.1 User Misdirects Configuration

The user is required to functionally test devices that are configured so user misdirection can be discovered during testing (refer to Section 7-1.1.3).

If the user configures a node unintentionally and it is not noticed, the originator connecting to the node will not have the matching signature and will either reconfigure the device with the correct signature or reject the connection. Either of these two actions will result in the system going to a safety state.

If the end user enables authentication features, users cannot configure nodes for which they do not have authorization. User authentication is an optional feature.



#### **7-1.6.1.2 System Misroutes configuration**

There are two different cases for a system to misroute a configuration: configuration preformed during connection establishment by a Type 1 safety open, and configuration from a workstation or Originator-to-Target configuration process. Both of these cases are explained in the following paragraphs.

- If a routing error occurs during the establishment of a configuration session the configuration mode message will be rejected by the device because of the wrong TUNID or OUNID in the mode message.
- If a configuration message is misrouted within a configuration session, it will be rejected by the receiving device if the device is not in the correct mode or is received on a connection other than the connection used to establish the configuration mode. If the device is in the correct mode the configuration message would have to be addressed to the correct class, instance, attribute to be accepted and would be discovered during the SCID validation steps. The device that was supposed to correctly receive the configuration message will also fail the SCID validation. The user will also discover this during the check of the configuration read back from the device.

#### **7-1.6.1.3 Lost Configuration Message**

The detections for a lost message are the same as those for a misrouted message.

#### **7-1.6.1.4 The Configuration Is Corrupted**

If for any reason the configuration is corrupted, it will be detected by the SCID checks, the user functional testing, and the user configuration examination.

#### **7-1.6.1.5 Configuration at an Inappropriate Time**

Configuration of operating devices is prevented by the user authentication and setting of the configuration mode. If a user attempts to configure a device that the user is not authorized for the mode request will be rejected. If the user sends configuration messages to a device without the configuration mode set the device will reject the changes.

#### **7-1.6.1.6 User Loads the Wrong Configuration**

Each configuration is uniquely identified by the SCID. Users are required to implement procedures to verify that the configuration being loaded is the correct one for the application.

#### **7-1.6.1.7 Configuration Process Is Interrupted**

If the configuration process is interrupted for any reason the device will remain in the configuration mode until a valid configuration is loaded and the mode is changed.

When the configuration process is interrupted the process will need to be restarted on a new connection with a new configuration request.

### **7-1.6.2 Detection Measures**

The following sections will explain each of the detection measures that are used by the SNCT interface for the detection of the errors during the download process.



#### **7-1.6.2.1 User Authentication**

All safety devices have optional password protection. The Configuration Lock, Configuration Request, Mode Change, and Reset commands all require the use of a password to change the state.

#### **7-1.6.2.2 Identification of Configuration File**

All safety configuration files are uniquely identified by the SCID See Section 7-1.3.1

#### **7-1.6.2.3 Identification of target to be configured**

The configuration mode message and the verify message both use the TUNID for identification of the target. See Section 7-1.3.5

#### **7-1.6.2.4 CRC protection on configuration**

Part of the SCID is the SCCRC. The SCCRC is 32 bit CRC that is calculated over the safety configuration data. See Section 7-1.3.2

#### **7-1.6.2.5 Configuration Ownership**

When an out-of-box target device is configured by an originator, the device captures the OUNID and subsequently prevents any other originator from configuring the device. Thus the ownership assignment (OUNID) provides a means of locking this device to a single configuration source. See Section 7-1.3.5

#### **7-1.6.2.6 Configuration Session Control**

Prior to configuring a device it must be placed into a configuration mode. The configuration mode is persistent in the device and remains in place until the device is set to another mode by a command. SRS33 The configuration mode in safety devices implementing the SNCT interface shall persist through power cycles.

SRS34 Upon entering the configuration mode, devices implementing the SNCT interface shall:

- Store the configuration mode in NVS
- Mark the existing configuration invalid
- Lock out all configuration messages for connections other than the one that set the configuration mode.

SRS35 To enter the configuration mode, devices implementing the SNCT interface shall:

- confirm User Password in the configuration mode command
- confirm TUNID in the configuration mode command

#### **7-1.6.2.7 User Functional Testing**

The user is required to functionally test new and replacement devices prior to utilizing the system in a SIL3 environment.



#### **7-1.6.2.8 Configuration Protection**

After configuration, testing, and user verification has been completed by one of the procedures defined in Section 7-1.5.1, all tool-configured devices are locked. These are Targets, Originators, or both. This locked state can be read to provide an indication that the configuration has been verified and also guards against inadvertent configuration changes by requiring the user to first unlock the device.

As a result of user testing, originator-configured target devices have ownership assignments which binds these devices to their owners; effectively locking these devices from being inadvertently configured by some other source. As shown in Figure 7-1.7, the combination of locked tool-configured originators and owned targets provides a secure environment to protect the verified system from unauthorized changes.

#### **7-1.7 Diagnostic Software Protections (Not Safety Related)**

The software may also implement data protection measures of the device configuration data as a data integrity measure and for the early error detection mechanism. It is important to note that these measures are not safety related and therefore do not need to achieve SIL3 or be certified. These areas are defined here:

Storage of Configuration Data to Hard Disk, CD, etc shall include the SCID and be checked during a file load operation.

The transfer of Configuration Data across software interfaces shall include the SCID and be checked by the receiving software component (refer to Section 7-1.1.3).

#### **7-1.8 Device Memory Architecture Considerations**

It is important to understand the memory architecture that target devices could use to know how it could behave during configuration. All these architectures are supported by this specification but will change the behavior for the user.

There are 3 distinct data areas defined with respect to configuration:

**NV stored copy** this is a valid configuration stored in Non-volatile memory and is what is used to set up a safety device on power-up.

**Working RAM copy** – This is a valid configuration that is an exact copy of the configuration in NV memory, but is what is used during run-time operation.

**Editable RAM Copy** – This is a data space that can be modified by software in the configuration mode, but has yet to be applied.

Product developers may choose to implement one or more of these data areas to increase user flexibility.

**NV Copy Only** This design approach keeps configuration only in NV memory. When the configuration mode is entered, the entire configuration is invalidated and there is no possibility to restore the configuration. The software must successfully download a complete and valid configuration to the device. Once it is validated and applied the NV copy is marked as good. The restore command is not supported in this type of device.



**NV and Working Copy** – This design approach keeps configuration in NV memory and a working copy in RAM. When the device enters the configuration state, the working copy is modified, but until the configuration is validated and applied, the old valid configuration is still available to be restored. A power cycle before the Apply would revert to the last configuration in NV memory. The restore command is supported and would re-copy the NV version into working RAM.

**NV and Working Copy with Edit buffers** – This design approach keeps a valid working configuration at all times until a “validate and apply” is done. If the configuring device quits without applying the changes, the restore command would allow the device to simply exit the configure state and run with what is in memory. The restore command is supported, but no copy from NV RAM is needed, the working copy can be used.



## **7-2 Electronic Data Sheets**

### **7-2.1 General Rules for EDS based Safety devices**

#### **7-2.1.1 Safety Configuration Assembly Definition**

Like most CIP devices, safety devices can be configured from an EDS file definition.

FRS349 Safety devices that are configured via EDS file shall define a Configuration Assembly so that the software knows how to calculate the SCID.

A standard EDS assembly section shall be used to define the various parameters contained in the Safety Configuration Assembly so the software can provide appropriately useful user prompts. A field and keyword has been defined in the [Parameter Class] section to identify the instance Id of the assembly. A connection manager entry shall also be used to reference this assembly if it is to be configurable with a Type1 SafetyOpen. This reference will publicize to the SNCT which originator connections are appropriate to configure the device.

#### **7-2.1.2 Configuration CRC**

The SNCT and EDS based safety device must agree on a method for calculating the SCCRC. EDS based safety devices shall use CRC-S4 with the seed value defined in Appendix E. The data input to the CRC algorithm is defined by the Configuration Assembly. Any padding in the configuration assembly will be assumed to be zero for purposes of calculating the SCCRC.

#### **7-2.1.3 Password Encryption for EDS Devices**

The SNCT and any other configuration software implementing the SNCT interface shall have a standardized method for encrypting passwords prior to presenting them to a safety device. Software tools implementing the SNCT interface shall use CRC-S4 with the seed value defined in Appendix E.

When an encrypted password is smaller than the fixed-size password parameter, the method described under the Set\_Password service in the Safety Supervisor object definition shall be used.

### **7-2.2 General EDS Extensions for Safety**

#### **7-2.2.1 Extension to [File] Section for Safety**

##### **7-2.2.1.1 EDS File CRC**

EDS Files for safety device shall be protected by a CRC. CRC-S4 with the seed value defined in Appendix E shall be used.

The EDS File CRC value shall be entered in the [file] section with the entry keyword "EDSFileCRC=". The EDSFileCRC entry keyword shall be contained on a single line.

The SNCT shall confirm the EDS file CRC prior to using the file contents as described below:

1. Open file as a binary file. Load the entire file into memory.
2. Remove the entire line including the carriage return that has the EDSFileCRC= entry.
3. Calculate the CRC over the remaining contents of the file.
4. Compare the calculated CRC to the EDSFileCRC= entry's value.



SRS207 An “EDSFileCRC=” keyword and value shall be included in all safety device EDS files. If the [Device Classification] entry describes the device as having any Safety capabilities and the EDSFileCRC= entry keyword does not exist, then the SNCT shall not use the file.

### 7-2.2.2 Extensions to [Device Classification] Section for Safety

SRS206 The EDS file for safety devices shall include a [Device Classification] section which shall contain a **Classx** keyword that indicates the Safety classification.

The Classx keyword, with the Classification field equal to Safety, is defined in Table 7-2.1.

**Table 7-2.1 Safety Classx Entry Format**

Field Name	Field Number	Data Type	Required/Optional
Classification	1	Field Keyword	Required
Reserved	2, 3	empty	
Port Instance	4, 5, ... n	UINT	Conditional <sup>1</sup>
1 Required if the device supports more than 1 CIP port			

- Classification shall be the value Safety.
- The Port Instance fields are the Port Object instances that support safety.

#### 7-2.2.2.1 Example

EDS snippet showing how a 2 port device, TCP (EtherNet/IP) and DeviceNet, indicates that Safety is only supported by the DeviceNet port.

```
[Port]
  Port1 = TCP,           $ Port type
           "ENet",       $ Port name
           "20 F5 24 01", $ Path to Port object, The TCP/IP object
           3;            $ Internal Port Number
  Port2 = DeviceNet,     $ Port type
           "DNet",       $ Port name
           "20 03 24 01", $ Path to Port object, The DNet object
           4;            $ Internal Port Number

[Device Classification]
  Class1 - DeviceNet;    $ DeviceNet class device
  Class2 = EtherNetIP;   $ EtherNet/IP class device
  Class3 - Safety,       $ Safety class device
  ''
  2;                    $ Safety supported on Port Instance 2,
                        $ the DeviceNet port
```



### 7-2.2.3 Extension to [ParamClass] Section for Safety

#### 7-2.2.3.1 Parameter Class SafetyCfgAssembly Keyword

Table 7-2.2 Parameter Class Keywords

Entry Name	Entry Keyword	Field Number	Data Type	Required/Optional
Max Instances	MaxInst	1	UINT	Required
Parameter Class Descriptor	Descriptor	1	WORD	Required
Safety Assembly Instance	SafetyCfgAssembly	1	UINT	Conditional <sup>1</sup>

<sup>1</sup> Required for Safety devices that are configured via EDS file

**SafetyCfgAssembly** – Identifies the configuration assembly object instance number for safety-related parameters in this device.

SRS208 A “SafetyCfgAssembly=” keyword entry shall be included for Safety devices that are configured via EDS file. This value shall also match an AssemN entry keyword specified in the [Assembly] section.

**Example:**

\$ This device has only one configuration assembly and all the parameters are safety-related

[Parameter Class]

MaxInst = 3;  
Descriptor = 0x0E;  
SafetyCfgAssembly= 4;

### 7-2.2.4 Extension to [Connection Manager] Section for Safety

There are four new keywords defined for safety that are described in this section. The new keywords are summarized in Table 7-2.3.

Table 7-2.3 New Connection Manager Section Keywords for Safety

Entry Name	Entry Keyword	Field Number	Data Type	Required/Optional
Maximum Safety Connections	MaxSafetyConnections	1	USINT	Optional
Maximum Safety Input Connections	MaxSafetyInputCnxns	1	USINT	Optional
Maximum Safety Output Connections	MaxSafetyOutputCnxns	1	USINT	Optional
Default Safety Connections	DefaultSafetyConnections			
	Connection Instance 1	1	UINT	Required
	Connection Instance n	n	UINT	Conditional
Safety Format Supported	SafetyFormat	1	USINT	Optional



#### **7-2.2.4.1 Max Safety Connections**

Safety devices will typically support a large number of possible assemblies that mix and match different combinations of I/O data and/or status. At the same time, these devices have a limit on how many safety I/O connections can be established at any one time. These keywords in the [Connection Manager] section define these limits. If no entry exists in the EDS file, no limits exist.

##### **7-2.2.4.1.1 Example1**

A device can support 4 safety connections, and they can be of any type

```
[Connection Manager]
MaxSafetyConnections = 4;
MaxSafetyInputCnxns = 4;
MaxSafetyOutputCnxns = 4;
```

##### **7-2.2.4.1.2 Example2**

A device can support 4 safety connections.

Up to 4 input connections, but only 1 output connection

```
[Connection Manager]
MaxSafetyConnections = 4;
MaxSafetyInputCnxns = 4;
MaxSafetyOutputCnxns = 1;
```

#### **7-2.2.4.2 Default Safety Connections**

There will often be a set of connections that are most commonly used. Product developers can optionally define a default set of connections that allow the user to quickly and easily add these connections to an originator configuration. The DefaultSafetyConnections entry keyword specifies which connection manager entries are these “default” connections.

##### **7-2.2.4.2.1 Example**

A device has one default input and output connection. The device’s EDS file has 8 possible connections defined, but the defaults are defined in connection1 and connection7. To inform the software of this, the following entry is made in the [Connection Manager] section.

```
[Connection Manager]
DefaultSafetyConnections= 1,7;
```

##### **7-2.2.4.2.2 Safety Format Support**

A device which supports the extended format must declare that support by entering this keyword in the Connection Manager section of the device’s EDS file. By making this declaration, connection configuration software algorithms can be enabled to configure a new format type attribute in the Connection Configuration object. When this keyword is not present in an EDS file, it will be assumed only the base format is supported (equivalent to SafetyFormat = 1). For more information, refer to Section 2-10 and Section 5-6.2.6. The possible options are:



Safety Format Support	Value	Semantic
Base Only	1	Supports Base Format only
Extended Only	2	Supports Extended Format only
Base and Extended	3	Supports both the Base and Extended Formats

For example, to make the declaration that both formats are supported, the following entry is made in the [Connection Manager] section.

```
[Connection Manager]
    SafetyFormat = 3;    $ supports both base and extended format
```

### 7-2.2.4.3 Changes and Additions to Connection Fields

A Connection manager entry shall be defined for each CIP addressable safety component to which a connection can be established. The safety connections are defined using the existing Connection Manager section defined in Volume 1, Chapter 7 with the following extensions. The SNCT shall use each entry to present the user with a set of connection options that can be connected to from an originator.

The following is the list of fields currently defined for the Connection Manager section.

**Table 7-2.4 Connection Manager Field Usage for Safety**

Field Name	Field Number	Data Type	Usage for Safety
Trigger and transport	1	DWORD	Class: shall be only 0 Trigger: shall be application Application type: producers shall be Input Only consumers shall be Exclusive Owner ----- examples: producers (clients): 0x02040001 consumers (server): 0x84040001
Connection parameters	2	DWORD	O=>T size: shall only be fixed T=>O size: shall only be fixed Bytes per slot: always ignored Real Time Transfer format: 5 = Safety Priority: always High Connection Type: Inputs: T=>O Multi-cast or Point to-point O=>T Point-to-point Outputs: O=>T & T=>O Point-to-Point ----- examples: Multi-cast Input (producer): 0x22645505 Single-Cast Output (consumer): 0x22445505
O=>T RPI	3	Param	Either leave blank or provide Param reference to define device capability
O=>T size	4	UINT or Param	If blank, will either assign the size from the associated ASSEM reference or the software will use the Time Coordination size. (See Volume 5, chapter 2-1.7.1.5)
O=>T format	5	Assem	For Safety output connections, this field shall indicate the Safety output assembly to which the connection is associated. For safety input connections, this field shall be empty



Field Name	Field Number	Data Type	Usage for Safety
T=>O RPI	6	Param	Either leave blank or provide Param reference to define device capability
T=>O size	7	UINT or Param	If blank, will either assign the size from the associated ASSEM reference or the software will use the Time Coordination size (See Volume 5, chapter 2-1.7.1.5)
T=>O format	8	Assem	For Safety input connections, this field shall indicate the Safety input assembly to which the connection is associated. For safety output connections, this field shall be empty
proxy config size	9	UINT or Param	Can optionally be used
proxy config format	10	Param or Assem	Can optionally be used
target config size	11	UINT or Param	Size of configuration assembly when device supports configuration via Type 1 SafetyOpen.
target config format	12	Param or Assem	This field can indicate which assembly will hold the Type 1 SafetyOpen configuration data.
Connection name string	13	Eds_Char Array	Unique name to allow a user to identify the application assembly to which this instance is associated
Help string	14	Eds_Char Array	User help message related to using the connection.
Data Path	15	Eds_Char Array	Complete path comprising of Config, Consumed, and Produced path. Null paths are inserted where no path exists (see possible formats below)
ASYNCL	16	USINT	This is a CIP Safety exclusive field.  Only applies to producing connections. Field should be empty for consuming connections. Used to calculate Network Reaction Time
Max Consumer Number <sup>1</sup>	17	USINT	This is a CIP Safety exclusive field.  When safety devices wish to define multi-cast connections and need to restrict the max number of consumers to a value less than the default maximum of 15, this field can define the product limit. If this field is empty, the SNCT shall always use the default value of 15 for the maximum number of multi-cast connections. This field can be left empty for single-cast connections. See section 2-4.5.2.5 – Max_Consumer Number for the definition of legal values for this field.

<sup>1</sup> This field is specified only for safety related specifications and shall either be empty or omitted for non-safety related connection manager instances.

#### 7-2.2.4.3.1 Trigger and Transport Field

Safety devices will define whether a connection is a producing (client) or consuming (server) connections by filling out this field. The other parts of the field will have fixed definitions as follows:

Class: always 0

Trigger: always application

Transport type: all 0

examples:

producers (clients): 0x0004 0001

consumers (server): 0x8004 0001



#### 7-2.2.4.3.2 Connection Parameter Field

Safety Devices define what types of connections are possible for a connection by using this field. Typically producing (inputs) connections can support T=>O multi-cast or point-to-point, and outputs will always only allow O=>T as single-cast. In both cases, the connection type in the opposite direction will always be point-to-point only. All the other fields will have fixed values. The following table provides the only valid values for Safety devices.

**Table 7-2.5 Connection Parameter Field Settings for Safety**

Connection Type	Connection Parameter Setting
Producing (client) input connection	0x22645505 or 0x22445505 (no multi-cast)
Consuming (server) output connection	0x22445505

#### 7-2.2.4.3.3 Data Path Field

Volume 1, section 3-5.5.1.10 defines a method to construct an application path in a Forward Open for 3 references: one for configuration, one for consumed data, and one for produced data. With the introduction of NULL paths for safety, there are some additional requirements for safety. Refer to Chapter 3, section 3-3.7 for more information.

#### 7-2.2.4.3.4 Safety ASYNC Field

This parameter is necessary to properly calculate the Network Reaction Time for a safety connection. This parameter is generally different for input connection and output connections but is design dependent. This field is required in the Connection Manager section for producing (client) connections. Devices which produce on input connections must define a Safety ASYNC value for those connections, and devices which produce on output connections (i.e. logic devices or scanners) must define a Safety ASYNC value for these connections. Any value provided on consuming (server) connections will be ignored by the software.

SRS49 If a Safety ASYNC parameter is not defined in a producing connection, the software shall use a worst-case default value of 1.

Safety ASYNC = 0 = synchronous

Safety ASYNC = 1 = asynchronous

#### 7-2.2.4.4 [Connection Manager] for Safety Example

```
$ Sample Electronic Data Sheet illustrating the Parameter Groups
Section
[File]
...

[Device]
...

[Device Classification]
...

[ParamClass]
  MaxInst = 2;
  Descriptor = 0x0E;
```



```

SafetyCfgAssembly- 1;

[Params]
Param1 -
    0,                $ first field shall equal 0
    6, "20 0F 24 01 30 01", $ path size, path
    0x0000,           $ descriptor
    2,                $ data type: 16-bit WORD
    2,                $ data size in bytes
    "Idle state",     $ name
    "",               $ units
    "User Manual p48", $ help string
    0, 2, 1,          $ min, max, default data values
    0, 0, 0, 0,       $ mult, dev, base, offset scaling not used
    0, 0, 0, 0,       $ mult, dev, base, offset link not used
    0;                $ decimal places not used
Param2 = 0, 6, "20 0F 24 02 30 01", $ path size, path
    0x0000, 2, 2,
    "Fault state", "", "User Manual p49",
    0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0;

[Assembly]
Revision - 2;
Assem1 = "Safety Configuration", "20 04 24 01 30 03",
    1,                $ size = 1 byte
    0,                $descriptor - assembly "not editable"
    ,,               $reserved
    4, Param1,        $4-bits
    3, Param2,        $3-bits
    1, ;

Assem2 - "Safety Inputs",
    "20 04 24 02 30 03", $can be read via explicit message
    1,                $ size - 1 byte
    0,                $descriptor - assembly "not editable"
    ,,               $reserved
    1, "20 08 24 01 30 03", $Discrete Input Point 1
    1, "20 08 24 02 30 03", $Discrete Input Point 2
    6, ;             $6 pad bits

Assem3 = "Safety Outputs",
    "20 04 24 03 30 03", $can be read via explicit message
    1,                $ size - 1 byte
    0,                $descriptor = assembly "not editable"
    ,,               $reserved
    1, "20 09 24 01 30 03", $Discrete Output Point 1
    1, "20 09 24 02 30 03", $Discrete Output Point 2
    6, ;             $6 pad bits

[Connection Manager]
MaxSafetyConnections - 2;
MaxSafetyInputCnxns = 1;
MaxSafetyOutputCnxns = 1;
DefaultSafetyConnections = 1,2;
SafetyFormat = 3;

Connection1 = $Input Connection Configuration

```



```

0x00040001,    $ trigger & transport
                $ class 0, application trigger, client
0x22645505,    $fixed sizes, safety formats
                $ O->T point-to-point,
                $ T->O multicast or point-to-point
                $ priorities high
,,,           $ Time Coordination parameters left blank
,,           $ RPI set at originator, size obtained from ASSEM
[Assem2],      $ produced data from assembly 2
,,           $ config part 1 (not used)
,[Assem1],     $ Type 1 config accepted on this connection,
                $ size comes from ASSEM
"Safety Input 1", $ connection name
"",           $ Help string
20 04 24 02 20 04 24 FF 20 04 24 03,
                $ Long format
                $ Config @ instance 2, Null @ instance 0xFF,
                $ Produced @ instance 03
1,            $ asynchronous interface
15;           $ 15 max consumers

```

```

Connection2 = $Output Connection Configuration
0x80040001,    $ trigger & transport
                $ class 0, application trigger, server
0x22445505,    $fixed sizes, safety formats
                $ O->T point-to-point,
                $ T->O point-to point
                $ priorities high
,,           $ RPI set at originator, size obtained from ASSEM
[Assem3],      $ consumed data to assembly 3
,,,           $ Time Coordination parameters left blank
,,           $ config part 1 (not used)
,[Assem1],     $ Type 1 config accepted on this connection,
                $ size comes from ASSEM
"Safety Output 1",$ connection name
"",           $ Help string
20 04 24 02 2C 03 2C FF,
                $ Compressed format
                $ Config @ instance 2, Consumed CP=03,
                $ Null CP=0xFF,
,;            $ ASYNC and Max consumers

```



## **Volume 5: CIP Safety**

# **Chapter 8: Physical Layer**

---



## **Contents**

8-1	Introduction.....	3
-----	-------------------	---



## **8-1 Introduction**

This chapter of the CIP Safety Networks specification contains additions to the Physical Layer that are safety specific. At this time, no such additions exist.



This page is intentionally left blank



## **Volume 5: CIP Safety**

# **Chapter 9: Indicators and Middle Layers**



## Contents

9-1	Indicators .....	3
9-1.1	CIP Safety Indicator Requirements.....	3
9-1.2	LED Indications for setting the device UNID.....	3
9-1.3	Module Status LED. ....	4
9-1.4	Indicator Warning .....	4
9-1.5	Network Status LED .....	4
9-2	Switches.....	5
9-2.1	Switch Behavior on CIP Safety.....	5
9-2.1.1	MACID Switches .....	5
9-2.1.2	MACID determination.....	5
9-2.1.3	Use Cases .....	8
9-2.2	Reset Switch.....	10



## 9-1 Indicators

Indicators assist maintenance personnel in quickly identifying a problem unit. This is accomplished through the consistent placement and presentation of indicators on DeviceNet products.

This section refers to Indicators that are visible to maintenance personnel. Visible means that:

- Indicators can be viewed without removing covers or parts from the equipment.
- Indicators can be seen in normal lighting.

Any labels and icons must be visible whether or not the Indicator is illuminated.

### 9-1.1 CIP Safety Indicator Requirements

CIP Safety devices are required to have module and network indicators.

Device developers are free to have additional indicators on their products if they so desire.

**Important:** SRS186 CIP Safety Device shall provide a separate Module Status LED and a Network Status LED. In this case, these indicators provide important additional safety-related information to the user that cannot be accomplished with a combined Net/Mod Status indicator.

**Important:** If there is a conflict between turning an LED on Red versus Green, the LED should be turned on Red.

### 9-1.2 LED Indications for setting the device UNID

The following Table shows the way the LED indications are used for setting the UNID.

**Table 9-1.1 LED Indications for Setting UNID**

Safety Supervisor State/Event	MOD LED	NET LED
Self-test	Flash Red/Green	Off
Waiting for TUNID	Flash Red/Green	Connection dependent
Waiting for TUNID Proposed UNID Rcvd	Flash Red/Green	Flash Red/Green 250ms/250ms Reuse DeviceNet Group 4 pattern
Configuring	Flash Red/Green	Connection dependent
Idle	Flashing Green	Connection dependent
Executing	Green	Connection dependent



### 9-1.3 Module Status LED

This bi-color (green/red) LED provides device status. It indicates whether or not the device has power and is operating properly. Table 9-1.2 defines the Module Status LED states. Table 9-1.2 shows the Module Status LED conditions for Safety Devices (which must use the Safety Supervisor Object). The states shown reflect the device states specified in the Identity Object. These states are also mapped to the states defined in Safety Supervisor Object.

SRS187 For safety devices which use the SNCT interface, the states of the Safety Supervisor shall be what determine the Module LED status.

**Table 9-1.2 Module Status LED**

For Safety Supervisor state:	LED is:	To indicate:
No Power	Off	There is no power applied to the device.
Executing	Green	The device is operating in a normal condition
Idle state	Flashing Green <sup>1</sup>	The Device is in the Idle or Standby State. Reference the Identity Object or Safety Supervisor Object <sup>2</sup>
Abort	Flashing Red <sup>1</sup>	Recoverable Fault
Critical Fault	Red	The device has an unrecoverable fault, may need replacing
Device Self Testing, Waiting_for_TUNID or Configuring <sup>2</sup>	Flashing Red-Green <sup>1</sup>	The Device is in Self Test or the Device needs commissioning due to configuration or UNID missing, incomplete or incorrect <sup>3</sup> . Reference the Identity Object or Safety Supervisor Object <sup>2</sup>

<sup>1</sup> For information on LED flash rates, refer to section 8-2.8, DeviceNet Spec Vol. 1

<sup>2</sup> The state of Safety Devices is determined by the Safety Supervisor Object

<sup>3</sup> An Unconfigured/unowned Safety Device should not be allowed to persist while connected to an active safety system.

### 9-1.4 Indicator Warning

SRS105 The User Safety Manual shall provide a warning that states “LEDs are NOT reliable indicators and cannot be guaranteed to provide accurate information. They should ONLY be used for general diagnostics during commissioning or troubleshooting. Do not attempt to use LEDs as operational indicators”.

### 9-1.5 Network Status LED

This bi-color (green/red) LED indicates the status of the communication link. For DeviceNet Safety devices, refer to chapter 6, section 6.2, Figure 6.1, *Network Access State Transition Diagram*, of the DeviceNet Specification to compare the Network Status LED to the Network Access State machine.



**Table 9-1.3 Network Status LED states.**

For Safety Supervisor state:	LED is:	To indicate:
Not Powered/Not On-line	Off	Device is not on line The device has not completed the Dup_MAC_ID test yet The device may not be powered, look at Module Status LED.
On-line, Not Connected	Flashing Green <sup>1</sup>	Device is on-line but has no connections in the established state. The device has passed the Dup_MAC_ID test, is on line, but has no established connections to other nodes. For a Group 2 Only device it means that this device is not allocated to a master For a UCMM capable device it means that the device has no established connections For Safety Devices, a connection may be established, but the Validator has not completed an initial Time Coordination exchange.
Link OK, On-line, Connected	Green	The device is on-line and has connections in the established state.
Connection Time-out	Flashing Red <sup>1</sup>	One or more I/O Connections are in the Timed-Out state
Critical Link Failure	Red	Failed communication device. The device has detected an error that has rendered it incapable of communicating on the network
Communication Faulted and Received an Identify Comm Fault Request – Long Protocol or Propose_TUNID service received	Flashing Red-Green	A specific Communication Faulted device. The device has detected a Network Access error and is in the Communication Faulted state. The device has subsequently received and accepted an Identify Communication Faulted Request - Long Protocol message. Safety devices use this indication to allow maintenance personnel to commission the UNID in a safety device.

<sup>1</sup> For information on LED flash rates, refer to section 8-2.8, DeviceNet Spec Vol. 1

## **9-2 Switches**

### **9-2.1 Switch Behavior on CIP Safety**

#### **9-2.1.1 MACID Switches**

The following algorithm can also be applied to the handling of the Baud Rate switch if implemented. When hardware switches are used for setting MACID and/or Baud Rate there are some differences in how these values are processed in safety devices. This section will outline the algorithm.

There is direct relationship between the MACID setting and the MACID portion of the Target UNID (TUNID) in the safety supervisor. Refer to the Safety Supervisor object for more information.

#### **9-2.1.2 MACID determination**

During device initialization, the MACID switches are read by the device firmware.

SRS106 The MACID attribute (if supported) in the DeviceNet object shall always reflect the value in use.



The following logic is applied.

- SRS110 When the switches are read, if the switches specify a valid MACID, (a value from 0 to 63 for MACID), this value shall be used as the MACID.
- SRS111 When the switches are read, if the specified MACID differs from the value stored in the DeviceNet object's MACID Attribute (NVRAM values), the appropriate attribute/s shall be updated and saved to NVRAM.
- When the switches are read, if the switches specify an invalid MACID, (a value greater than 63), the device should follow one the recommended options in the DeviceNet standard.
- SRS113 When the switches are read, if the switches used to specify the MACID are set to a valid value (a value from 0 to 63 for the MACID), the MACID attribute shall have Get Only access.
  - SRS114 When the switches are read, if the switches are set to the software programmable mode (>63 for the MACID), the MACID attribute shall have Set access.
- SRS115 If a device has no switches it shall use default values for MACID in its out-of-box configuration (63 for the MACID) .
- SRS121 If the device is out-of-box it shall use the current MACID switch settings if these are valid values

FRS350 Safety devices shall monitor and detect changes in the MACID or the IP Attribute and Baud Rate (if applicable) switch settings. Developers are free to determine an appropriate sample period for this detection.

FRS351 When a MACID switch setting change is detected, the device shall update the MACID\_Switch\_Value attribute. If the MACID switch equals the MACID in use or the MACID Switches are greater than 63 and software settable clear the MACID changed attribute and make no change in the state of the device, otherwise set the MACID changed attribute and transition to the ABORT state. (refer to flow chart in Figure 9-2-1)

FRS364 When an IP switch setting change is detected it shall determine if the change creates a mismatch condition. If the IP switch differs from the IP Address attribute, it shall transition to the ABORT state.

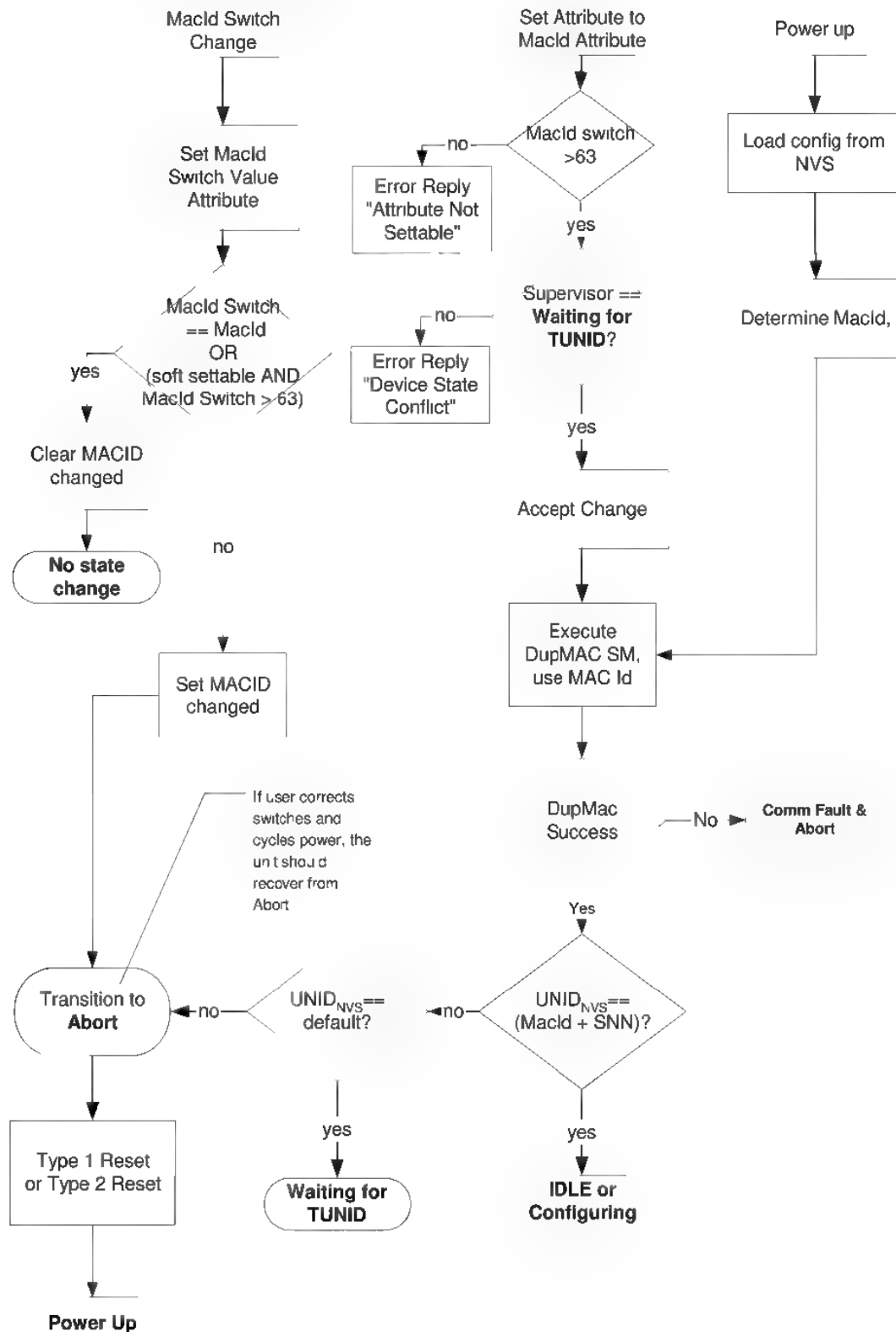
FRS354 When a Baud Rate switch setting change is detected, the device shall update the Baud\_Rate\_Switch\_Changed attribute and the Baud\_Rate\_Switch\_Value attribute, then transition to ABORT.

FRS352 If a CIP safety device supports the Set\_attribute service to the MACID or IP address attribute, and the MACID or IP address switches are set to enable the set service, Safety devices shall only accept the Set\_Attribute while in the "Waiting for TUNID" state. Otherwise, it shall reply with a "Device State Conflict".

FRS353 If a CIP safety device supports the Set\_attribute service to the Baud Rate attribute, Safety devices shall only accept the Set\_Attribute while in the "Waiting for TUNID" state. Otherwise, it shall reply with a "Device State Conflict".



Figure 9-2-1 Safety Device MACID Processing Logic





### **9-2.1.3 Use Cases**

**CASE 1: Initial Power on, MACID switches indicate valid position, UNID is at default**

1. The Target UNID value stored in NVS contains an invalid Number (default (**REQ**))
2. User sets MACID switches.
3. User powers up device
4. Device reads valid switch settings.
5. Device uses the settings and updates the MACID in the DeviceNet object (NVRAM). The MACID attribute is made “gettable” only.
6. Device Execute DupMac check with current MACID
7. Device reads Target UNID attribute (Safety Supervisor) and updates SNN attribute in DeviceNet object with SNN portion of UNID.
8. Device detects an default Target UNID
9. Device transits to “waiting for TUNID” (refer to Figure 9-2-1)
10. Since the device has no Target UNID, the first connection request the device will respond with a “UNID Not Set” error.
11. The originator responds with a Propose\_TUNID service.
12. Device compares the MACID portion of the Target UNID to the MACID value in the DeviceNet object
13. If the proposed Target UNID matches the MacId values, it saves the proposed Target UNID into the Proposed\_TUNID attribute, begins flashing the NET LED with the red/green pattern, and replies with success.
14. The originator (based on acknowledge controls) sends an Apply\_TUNID that matches the Proposed\_TUNID attribute.
15. The device stops flashing the NET LED, saves the UNID to NV memory, saves the SNN portion to the SNN attribute in the DeviceNet object and resets the Proposed\_TUNID to default. This update shall be done with safety integrity.
16. Device transits to “Configuring” mode (device is not configured)
17. Since the device is out-of-box, on the next connection request the device will respond with a Signature mismatch.
18. Device is configured (Tool or Type 1 SafetyOpen)
19. The completion of the Type 1 SafetyOpen or the Apply service that completes the configuration.
20. Device is ready for normal operation

**CASE 2: Initial Power on, MACID switch in position >63, UNID is at Default**

1. The Target UNID value stored in NVS contains an invalid Number (default)
2. The MACID attribute in the DeviceNet object represent default value (**REQ**)
3. MACID switches do not represent valid address (>63)
4. User powers up device
5. Device reads switches and recognizes invalid address
6. Device reads Target UNID attribute (Safety Supervisor) and updates Safety Network Number attribute in DeviceNet object with SNN portion of Target UNID.
7. Device detects the default Target UNID, so the device **uses** the default MACID. This operation shall be done with safety integrity.
8. Device uses default value (63) and executes NASM (DUPMAC check)
9. Device transits to “waiting for TUNID” (refer to Figure 9-2-1)
10. User sets MACID with Set\_Attribute\_Single() service. Address is stored to DeviceNet object attributes
11. Device executes NASM (DUPMAC check) with new value



12. Device UNID- MACID comparison check. Since the UNID is at default, the device goes to “Waiting for TUNID” (refer to Figure 9-2-1)
13. Since the device has no Target UNID, the first connection request the device will respond with a “UNID Not Set” error.
14. The originator responds with a Propose\_TUNID service.
15. Device compares the MACID portion of the Target UNID to the MACID value in the DeviceNet object
16. If the proposed UNID matches the MACID, it saves the proposed UNID into the Proposed\_TUNID attribute, begins flashing the NET LED with the red/green pattern, and replies with success.
17. The originator (based on acknowledge controls) sends an Apply\_TUNID that matches the Proposed TUNID attribute.
18. The device stops flashing the NET LED, saves the Target UNID to NV memory, save the SNN portion in the Safety Network Number attribute of the DeviceNet object, and resets the Proposed\_TUNID to default. This update shall be done with safety integrity.
19. Device transits to configure mode (out-of-box condition)
20. Upon the next connection request the device will respond with an error (Signature mismatch)
21. Device is configured (Tool or Type 1 SafetyOpen)
22. The completion of the Type 1 SafetyOpen or the Apply service that completes the configuration will store the device configuration. This update shall be done with safety integrity.

**CASE 3: Modification of MACID Switch.**

**Unit is operating, MACID switch in range of 0-63, UNID has a non-default setting and MACID matches the UNID value.**

1. User sets MACID to new position
2. Device detects that the MACID switch has changed and the switches are in range 0-63
3. DeviceNet object MACID\_Switch\_Changed and MACID\_Switch\_Value attributes are updated.
4. Device transitions to ABORT state, MOD LED changed to flashing red.
5. device continues to use the "old" MACID

**CASE 4: Set\_attribute service changes the MACID while operating normally.**

**Unit is operating, MACID switch in position >63, UNID has a non-default setting and MACID matches the UNID value.**

1. Device is in a state other than “Waiting for TUNID”
2. MACID switches set to address >63
3. User attempts to set MACID with Set\_Attribute\_Single() service.
4. Device sends error reply with error code “Device State Conflict” (refer to Figure 9-2-1)



**CASE 5: Set\_attribute service changes the MACID while device is “Waiting for TUNID”. MACID switch in position >63, MACID in NVRAM has valid value, UNID value is at default.**

1. Device powered up using the MACID stored in NVRAM
2. UNID was at default, so device enters “Waiting for TUNID”
3. MACID switches set to address >63
4. User sets MACID with Set\_Attribute\_Single() service. Address is stored to DeviceNet object attributes
5. Device executes NASM (DUPMAC check) with new value
6. Device UNID- MACID comparison check. Since the UNID is at default, the device remains in “Waiting for TUNID” (refer to Figure 9-2-1)
7. UNID is set via Propose/Apply using the new MACID value
8. Device transits to CONFIGURING mode

### **9-2.2 Reset Switch**

SRS126 If a reset switch is provided on a safety device, the safety device shall execute a Safety Supervisor Type 2 reset (i.e. reset to out-of-box, but preserve the password).



## **Volume 5: CIP Safety**

# **Chapter 10: Bridging and Routing**



## **Contents**

10-1	Bridging Requirements for Safety .....	3
10-1.1	Bridged Multi-Cast Connections .....	3
10-1.2	Connections Originating on Ethernet or other non-DeviceNet CIP networks .....	5
10-1.3	Connections Originating on DeviceNet to Off-Link Targets .. .....	6
10-1.4	Multi-cast Connections Originating and Terminating on DeviceNet .....	7
10-1.5	Multi-cast Connections Originating and Terminating on non-DeviceNet CIP Networks .....	8



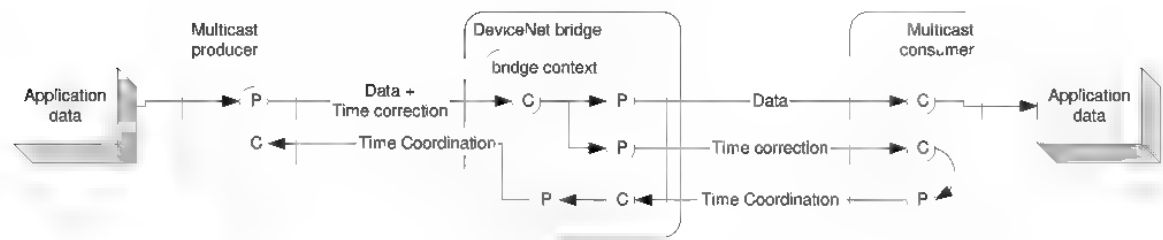
## 10-1 Bridging Requirements for Safety

### 10-1.1 Bridged Multi-Cast Connections

When multi-cast connections are bridged between DeviceNet and other Non-DeviceNet networks, the safety-aware DeviceNet bridge shall translate the connections from three messages on DeviceNet to two connections on Non-DeviceNet Networks.

The safety-aware DeviceNet bridge shall execute the 3-to-2 multi-cast safety connection translation by concatenating the safety data message and the time correction message at the bridge. Figure 10-1.1 shows the mapping of producers and consumers in a bridge.

**Figure 10-1.1 Non-DeviceNet to DeviceNet Bridge Example**



A non-certified DeviceNet Bridge needs to implement the following requirements when translating the single connection Data Message consisting of a Data&Time\_Stamp section, and a Time\_Correction section into a Data Message connection consisting of a Data&Time\_Stamp section, and a separate Time\_Correction Message connection.

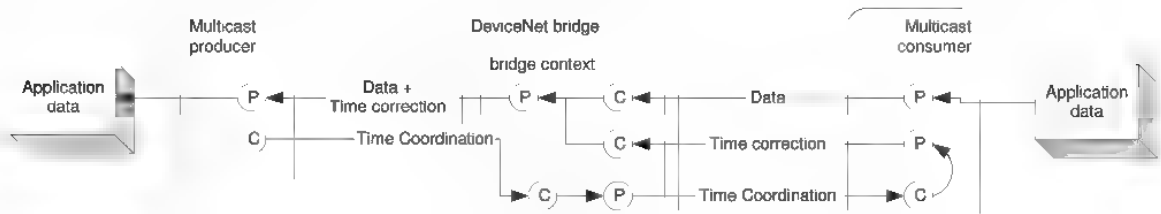
A non-certified DeviceNet Bridge, bridging a multi-cast production from a Non-DeviceNet network to a DeviceNet network shall forward a Data Message consisting of a Data&Time\_Stamp section onto the DeviceNet network, once and only once, any time a Data Message is received from the Non-DeviceNet network consisting of a Data&Time\_Stamp section, and a Time\_Correction section.

A non-certified DeviceNet Bridge, bridging a multi-cast production from a Non-DeviceNet network to a DeviceNet network shall forward a Time\_Correction Message onto the DeviceNet network, once and only once, any time a Data Message is received from the Non-DeviceNet network with a Time\_Correction section - MCast\_Byte.Consumer\_Num equal to a value other than 0x0. If the MCast\_Byte.Consumer\_Num is equal to 0x0 (Null Time\_Correction section), the Time\_Correction Message shall not be forwarded.

The non-certified DeviceNet Bridge shall not translate a Time\_Coordination Message. It is forwarded upon reception.



**Figure 10-1.2 DeviceNet to Non-DeviceNet Bridge Example**



A non-certified DeviceNet Bridge should use the following logic when translating the two Data Message and Time\_Correction Message connections into a single Data Message connection. The single Data Message connection contains a Data&Time\_Stamp section and a Time\_Correction section.

A non-certified DeviceNet Bridge, bridging a multi-cast Data Message and Time\_Correction Message from a DeviceNet network to a Non-DeviceNet network shall send a Data Message consisting of a Data&Time\_Stamp section and a Time Correction section onto the Non-DeviceNet network any time it receives a Data Message consisting of a Data&Time\_Stamp section from the DeviceNet network.

Every Time\_Correction Message or Messages received from the DeviceNet network shall be forwarded once and only once in order, in the Time Correction section of the Non-DeviceNet Data Messages that are triggered by the reception of the DeviveNet Data Message.

When sending a Data Message onto the Non-DeviceNet network, if there are no Time\_Correction Messages to be forwarded, the previous Time\_Correction section shall be sent with the MCast\_Byte.Consumer\_Num equal to 0x0 (Null Time\_Correction section).

For the case of the first Data Message being sent onto Non-DeviceNet before a DeviceNet Time\_Correction Message has been received, a Time\_Correction section of all 00's shall be sent.

The DeviceNet Bridge shall never create or modify Safety CRCs for the Data&Time\_Stamp, Time\_Correction, or Time\_Coordination sections.

The last router/bridge in a multi-hop connection shall convert the Forward\_Open requests received from a higher-level network into the equivalent Safety Open request when the target is on DeviceNet.

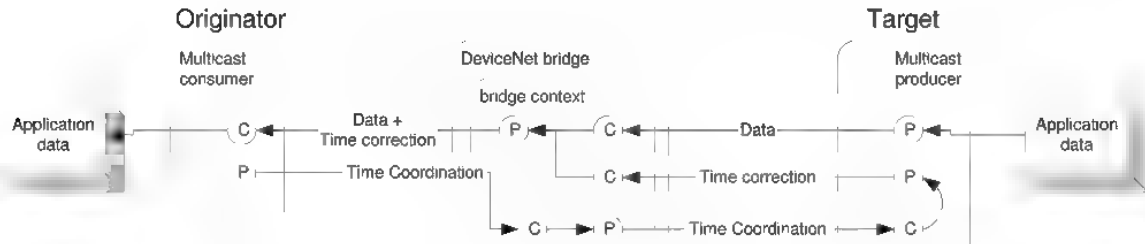
When processing a SafetyClose, if an intermediate node cannot find the connection that is to be closed (it may have timed out at the node), the SafetyClose request shall still be forwarded to downstream nodes or the target application.



## 10-1.2 Connections Originating on Ethernet or other non-DeviceNet CIP networks

When the Originator of the connection, shown on the left in Figure 10-1.3, is on Ethernet, the processing is as follows.

Figure 10-1.3 Connection Origination on Ethernet or other non-DeviceNet CIP networks



- The Software **must know that** one of the nodes for this connection is on DeviceNet **AND that the connection is of type Multi-cast**
  - It then knows that the 3<sup>rd</sup> connection point information must be filled out
    - Time Correction RPI value entered (a calculated multiple of the Data RPI)
    - Default parameters filled in
  - The standard **T-to-O** connection type set to Multi-cast
    - The T-to-O connection size is = Data size + Time Correction size (6)
  - The standard **O-to-T** connection type set to point-to-point
    - The O-to-T connection size is = Time Coordination size (6)
- The originator does not need to do any size adjustments. The sizes for the standard connections are correct.
- The bridge, upon receiving the ForwardOpen that contains a Safety Segment must
  - Recognize that the Connection request is a “SafetyOpen”
  - Open a Class 3 connection to the target node
  - **Recognize that the T-to-O connection type is a Multi-cast**
  - Subtract the 3<sup>rd</sup> connection point size from the standard T-to-O size and allocate resources for two consumer connections appropriately.
  - Allocate the resources for the single producer connection
  - Send the SafetyOpen over the class 3 connection
    - (initial request) Modify the Connection IDs to request all IDs be assigned by the target
  - Set up the IDs for the connections based on the values returned from the target.
- The target, upon receiving the SafetyOpen must
  - Accept the Class 3 connection request
  - Route the SafetyOpen message to the Connection object via the Message Router
  - Analyze the Connection Parameters and **recognize that this is a Multi-cast request**
  - Error checks the T=>O, O=>T, and TC connection parameters
  - Verifies that connection IDs requested can be provided
  - Subtracts the 3<sup>rd</sup> connection point size from the standard T-to-O size and allocates resources for two producer connections appropriately. Connection IDs assigned from available resources (if possible).

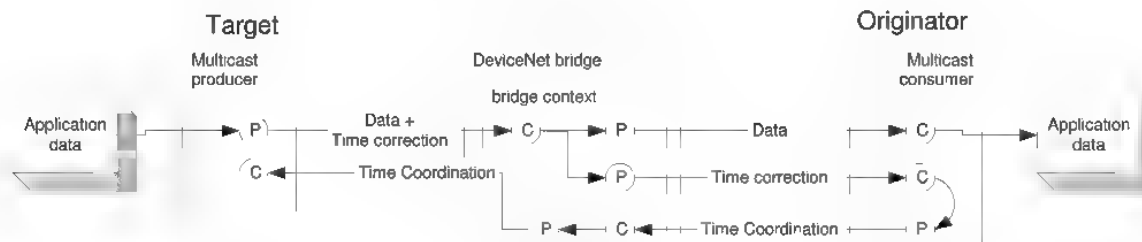


- Allocate the resources for the single consumer connection. Connection IDs assigned from available resources (if possible).
- Allocates an available consumer\_number
- Sends the SafetyOpen response with selected consumer\_number, connection IDs, and PID.

### 10-1.3 Connections Originating on DeviceNet to Off-Link Targets

When the Originator of the connection, shown on the right in Figure 10-1.4, is on DeviceNet, the processing is as follows:

Figure 10-1.4 Connection Origination on DeviceNet



- The Software must know that one or more of the nodes for this connection is on DeviceNet AND that the connection is of type Multi-cast
  - It then knows that the 3<sup>rd</sup> connection point information must be filled out
    - Time Correction RPI value entered (a calculated multiple of the Data RPI)
    - Default parameters filled in
  - The standard **T-to-O** connection type set to Multi-cast
    - The T-to-O connection size is = Data size + Time Correction size (6)
  - The standard **O-to-T** connection type set to point-to-point
    - The O-to-T connection size is = Time Coordination size (6)
- The originator must
  - **Parse & analyze the connection path** (provided by the software) to determine the target is off-link and that a SafetyOpen will be sent to an intermediary node (i.e. bridge).
  - Open a Class 3 connection request to the **bridge MacId** (obtained from the path)
  - **Recognize that the T-to-O connection type is Multi-cast**
  - Subtract the 3<sup>rd</sup> connection point size from the standard T-to-O size and allocate buffers for two consumer connections appropriately.
  - Allocate the resources for the single producer connection
  - Assign the IDs for the single producer connection based on ID resources available, set this value in the O-to-T connection Id of the ForwardOpen.
  - Format and send a ForwardOpen explicit message to the Connection Manager object in the bridge node
  - Set up the IDs for the two consumer connection based on the values returned by the bridge.
- The bridge, upon receiving the ForwardOpen must
  - Recognize that the Connection request message is a ForwardOpen with a Safety Segment in the path

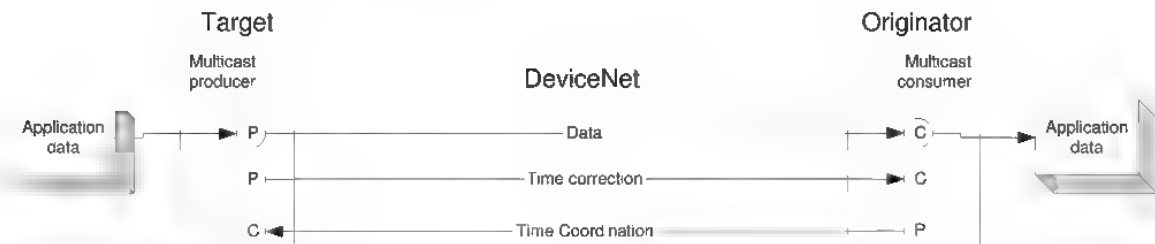


- **Recognize that the T-to-O connection type is a Multi-cast**
- Subtract the 3<sup>rd</sup> connection point size from the standard T-to-O size and allocate buffers for two producer connections appropriately (if they don't already exist from a previous connection).
- Allocate the resources for the single consumer connection
- Forward the message as an unconnected ForwardOpen
- Allocate the IDs for the two producer connections based on available resources and return them to the originator (or provide the IDs already assigned).
- Set the consumer ID based on the value from the ForwardOpen
- Forward the connection request to the Target device
- Set up these resources based on the sizes of the standard O-to-T and T-to-O connections.
- The target, upon receiving the "ForwardOpen" must
  - Route the message to the Connection Manager
  - Analyze the Connection Parameters and **recognizes that it is a Multi-cast request**
  - Error checks the T=>O, O=>T connection parameters
  - Allocates the resources based on the O-to-T and T-to-O sizes (the 3<sup>rd</sup> connection point information is ignored), and allocates resources for single producer connection (if they don't already exist from a previous connection).
  - Allocate the resources for the single consumer connection. Connection IDs assigned from available resources (if possible).
  - Allocates an available consumer number
  - Sends the ForwardOpen response with selected consumer number, connection IDs, and PID.

#### 10-1.4 Multi-cast Connections Originating and Terminating on DeviceNet

When both the originator and target are on DeviceNet, the following process is followed.

Figure 10-1.5 Local DeviceNet Connections



- The Software **must know** that one or more of the nodes for this connection is on DeviceNet **AND that the connection is of type Multi-cast**
  - It then knows that the 3<sup>rd</sup> connection point information must be filled out
    - Time Correction RPI value entered (a calculated multiple of the Data RPI)
    - Default parameters filled in
  - The standard **T-to-O** connection type set to Multi-cast
    - **The T-to-O connection size is = Data size + Time Correction size (6)**
  - The standard **O-to-T** connection type set to point-to-point
    - The O-to-T connection size is = Time Coordination size (6)

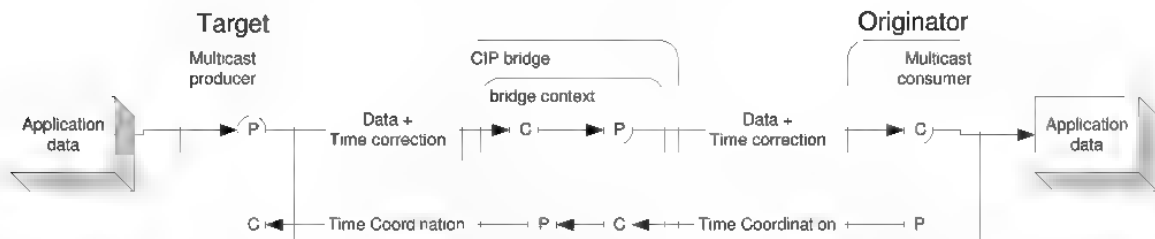


- The DeviceNet originator must
  - **Parse & analyze the connection path** (provided by the software) to determine the target is on-link and that a SafetyOpen is sent directly to the target node.
  - Open a Class 3 connection request to the **target MacId**
  - **Recognize that the T-to-O connection type is a Multi-cast**
  - Subtract the 3<sup>rd</sup> connection point size from the standard T-to-O size and allocate buffers for 2-consumer connections appropriately.
  - Allocate the resources for the 1-producer connection
  - Assign the Ids for the 1-producer connection based on ID resources available, set this value in the O-to-T connection Id of the SafetyOpen.
  - Format and send a SafetyOpen explicit message to the Connection object in the target node
  - Set up the Ids for the 2-consumer connection based on the values returned by the target.
- The target node, upon receiving the SafetyOpen will
  - Route the message to the Connection object
  - Recognize a SafetyOpen message **with the T-to-O connection type Multi-cast**
  - Subtract the 3<sup>rd</sup> connection point size from the standard T-to-O size and allocate buffers for 2-producer connections appropriately (if they don't already exist from a previous request).
  - Allocate the resources for the 1-consumer connection
  - Allocate the Ids for the 2-producer connections based on available resources and return them to the originator (or provide the Ids from the existing connections)
  - Set the consumer Id based on the value from the SafetyOpen
  - Reply to the connection request with the chosen Network Ids
  - Allocates an available consumer number
  - Sends the ForwardOpen response with selected consumer number, connection IDs, and PID.

### 10-1.5 Multi-cast Connections Originating and Terminating on non-DeviceNet CIP Networks

When both the originator and target are on non-DeviceNet networks, the following process is followed. These connections may or may not be through bridges or routers, but the result is the same.

Figure 10-1.6 Multi-cast Connections on non-DeviceNet CIP Networks



- The Software **must know** that one or more of the nodes for this connection is **not** on DeviceNet



- It then knows that the 3<sup>rd</sup> connection point information can be set to default values
  - Default parameters filled in
- The standard **T-to-O** connection type set to Multi-cast
  - The T-to-O connection size is = Data size + Time Correction size (6)
- The standard **O-to-T** connection type set to point-to-point
  - The O-to-T connection size is = Time Coordination size (6)
- The CIP originator
  - The originator does not need to do any size adjustments. The sizes for these connections are always correct.
  - Allocate the resources for the single producer connection
  - Assign the IDs for the single producer connection based on ID resources available, set this value in the O-to-T connection ID of the ForwardOpen.
  - send the ForwardOpen message to the target node
  - Set up the IDs for the consumer connection based on the values returned by the target.
- The target node, upon receiving the ForwardOpen will
  - Route the message to the Connection Manager
  - Recognize a T-to-O connection type Multi-cast
  - The target does not need to do any size adjustments. The sizes for these connections are always correct.
  - Allocate the resources for the single consumer connection
  - Allocate the IDs for the two producer connections based on available resources and return them to the originator (or provide the IDs from the existing connections)
  - Set the consumer ID based on the value from the ForwardOpen
  - Allocates an available consumer number
  - Sends ForwardOpen response with selected consumer number, connection IDs, and PID.



This page is intentionally left blank



## **Volume 5: CIP Safety**

# **Appendix A: Explicit Messaging Services**



## **Contents**

A-1	Introduction.....	3
-----	-------------------	---



## **A-1 Introduction**

This chapter of the CIP Safety Networks specification contains additions to CIP Engineering Units that are safety specific. At this time, no such additions exist.



This page is intentionally left blank



## **Volume 5: CIP Safety**

# **Appendix B: Status Codes**

---



## **Contents**

B-1	Introduction .....	3
-----	--------------------	---



## **B-1 Introduction**

This chapter of the CIP Safety Networks specification contains additions to the Status Codes that are safety specific. At this time, no such additions exist.



This page is intentionally left blank



## **Volume 5: CIP Safety**

# **Appendix C: Data Management**

---



## **Contents**

C-1	Introduction.....	3
C-2	Safety Network Segment .....	3
C-3	Safety Network Segment: Target Format (0x00) .....	3
C-4	Safety Network Segment: Router Format (0x01).....	5
C-5	Safety Network Segment: Extended Format (0x02) .....	5



## C-1 Introduction

This chapter of the CIP Safety Networks specification contains additions to the Data Management specifications that are safety specific.

## C-2 Safety Network Segment

The Safety Connection Parameters Segment Type has been added to the CIP Network Segment definition as segment type 0x10 (resulting in a segment value of 0x50). The updated Network Segment definition is shown in Table C-2.1 below.

**Table C-2.1 Safety Network Segment Identifier**

Network Segment Type	Network Segment Name
0x50	Safety segment

The format of the Safety Network Segment is shown in Table C-2.2. This segment type is in the range of Network Segment types, which contains a data size field. The Safety Segment Format parameter, the first parameter in the data area, indicates what data follows.

**Table C-2.2 Safety Network Segment Definition**

Byte Offset	Field Name	Data Type	Description
0	Segment Type	BYTE	Indicates Safety Network Segment (0x50)
1	Network Segment Data Length	BYTE	Length of segment data to follow (in words)
2	Safety Network Segment Format	USINT	Indicates type of Safety Network Segment 0 = Target format 1 = Router format 2 = EF Format 3 – 255 = Reserved
3 thru n	Safety Network Segment Data	ARRAY of octet	Safety Network Segment data, defined by segment format.

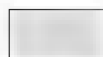
## C-3 Safety Network Segment: Target Format (0x00)

The Target Format, shown in Table C-3.1, is directed at the target of the connection. The overall segment size for the target format is 56 bytes or 28 words. The “Network Segment Data Length” parameter is 27 words (0x1B).



**Figure C-3.1 Safety Network Segment: Target Format (0x00)**

Field Name	Data Type	Description
Reserved	Byte	Pad
Configuration CRC (SCCRC)	UDINT	CRC-S32 of target Config Data Segment. A value of 0 is an indicator that SCID check can be skipped.
Configuration TimeStamp (SCTS)	DATE_AND_TIME	Time and Date stamp of the configuration. A value of 0 is an indicator that SCID check can be skipped. 6-byte value (Date: 2-bytes, Time: 4-bytes)
Time Correction EPI	UDINT	Not used: 0 When used: Time Correction Connection Instance:: EPI
Time Correction Network Connection Parameters	WORD	Network Connection Parameters for the Time Correction connection. This field uses the same definition as the Network Connection Parameter within the Connection Manager object. When not used: 0 (Null) When used: Fixed, High Priority, Multi-cast
Target_UNID (TUNID)	Struct UINT 10: octets	UNID of the Target device. This value is a 10-byte value consisting of the Safety Network Id (6 octets) and the Node Address (4-byte address)
Originator UNID (OUNID)	Struct UINT 10: octets	UNID of the Originator device. This value is a 10-byte value consisting of the Safety Network Id (6 octets) and the Node Address (4-byte address)
Ping_Interval_EPI_Multiplier	UINT	Number that defines the Ping_Count_Interval for the connection. Valid range of values is give by FRS235
Time_Coord_Msg_Min_Multiplier	UINT	For Producing Connection Instance. Valid range is 0 to 7813.
Network_Time_Expectation_Multiplier	UINT	For Consuming Connection Instance. Valid range is 0 to 42969
Timeout_Multiplier	USINT	Used by producer and consumer to timeout connections. Valid range 1 to 4
Max_Consumer_Number	USINT	Value: 1 (Single-Cast), 2-15(Multi-cast)
Connection Parameters CRC (CPCRC)	UDINT	CRC-S32 of target connection parameters in this request.
Time Correction Connection ID	UDINT	Not Used. 0xFFFFFFFF When used: Time Correction Connection Instance:: Connection ID



Safety Segment Fields  
covered by Connection  
Parameters CRC



## C-4 Safety Network Segment: Router Format (0x01)

The Router Format, shown in Table C-4.1, is directed at an intermediate router along the connection path when that router needs to know this is a Safety Forward\_Open. Currently, only DeviceNet routers need this network segment directed to them. The Network Segment is applied to the subnet from which the connection request originated. Router Format Safety Network Segments will only appear in the Path portion of the Connection\_Path of a Safety ForwardOpen. The overall segment size for the router format is 14 bytes or 7 words. The “Network Segment Data Length” parameter is 6 words.

**Figure C-4.1 Safety Network Segment: Router Format (0x01)**

Field Name	Data Type	Description
Reserved	Byte	Reserved, value = 0
Time Correction Connection ID	UDINT	Not Used: 0xFFFFFFFF When used: Time Correction Connection Instance:: Connection ID
Time Correction EPI	UDINT	Not used: 0 When used: Time Correction Connection Instance:: EPI
Time Correction Network Connection Parameters	WORD	Network Connection Parameters for the Time Correction connection This field uses the same definition as the Network Connection Parameter within the Connection Manager object. Not used: 0 (Null) When used: Fixed, High Priority, Multi-cast

## C-5 Safety Network Segment: Extended Format (0x02)

The EF Format is directed at the target of the connection and shown in the table below.

FRS373 All EF Format safety connections shall be established using the Extended format safety network segment in a Forward\_Open (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet).

Field Name	Data Type	Description
Reserved	Byte	Pad
Configuration CRC (SCCRC)	UDINT	CRC-S32 of target Config Data Segment. A value of 0 is an indicator that SCID check can be skipped.
Configuration TimeStamp (SCTS)	DATE_AND_TIME	Time and Date stamp of the configuration. A value of 0 is an indicator that SCID check can be skipped 6-byte value (Date: 2-bytes, Time: 4-bytes)
Time Correction EPI	UDINT	Not used: 0 When used: Time Correction Connection Instance:: EPI
Time Correction Network Connection Parameters	WORD	Network Connection Parameters for the Time Correction connection. This field uses the same definition as the Network Connection Parameter within the Connection Manager object When not used: 0 (Null)



Field Name	Data Type	Description
		When used: Fixed, High Priority, Multi-cast
Target_UNID (TUNID)	Struct UINT 10: octets	UNID of the target device. This value is a 10-byte value consisting of the Safety Network Id (6 octets) and the Node Address (4-byte address)
Originator UNID (OUNID)	Struct UINT 10: octets	UNID of the Originator device. This value is a 10-byte value consisting of the Safety Network Id (6 octets) and the Node Address (4-byte address)
Ping_Interval_EPI_Multiplier	UINT	Number that defines the Ping_Count_Interval for the connection. Valid range of values is give by FRS235
Time_Coord_Msg_Min_Multiplier	UINT	For Producing Connection Instance. Valid range is 0 to 7813.
Network_Time_Expectation_Multiplier	UINT	For Consuming Connection Instance. Valid range is 0 to 42969
Timeout_Multiplier	USINT	Used by producer and consumer to timeout connections. Valid range of 1 to 255 for HiAvailabilityFormat
Max_Consumer_Number	USINT	Value: 1 (Single-Cast), 2-15(Multi-cast)
Max_Fault_Number	UINT	Used by both producers and consumers to determine how many erroneous packets can be dropped before a connection must be closed  Default = 5.  A value of 0 indicates connections must be closed on any detected error.
Connection Parameters CRC (CPCRC)	UDINT	CRC-S32 of target connection parameters in this request.
Time Correction Connection ID <sup>1</sup>	UDINT	Not Used. 0xFFFFFFFF  When used: Time Correction Connection Instance:, Connection ID
Initial Time Stamp <sup>1</sup>	UINT	Used by Consumer to initialize Last_Time_Stamp_for_Rollover  Consuming Originator sets to 0xFFFF
Initial Rollover Value <sup>1</sup>	UINT	Used by Consumer to initialize TS_Rollover_Count  Consuming Originator sets to 0xFFFF

<sup>1</sup> These fields are NOT included in the CPCRC.



## **Volume 5: CIP Safety**

# **Appendix D: Engineering Units**

---



## **Contents**

D-1	Introduction.....	3
-----	-------------------	---



## **D-1 Introduction**

This chapter of the CIP Safety Networks specification contains additions to CIP Engineering Units that are safety specific. At this time, no such additions exist.



This page is intentionally left blank



## **Volume 5: CIP Safety**

# **Appendix E: Safety CRCs**

---



## **Contents**

E-1	Introduction.....	3
E-2	Native CRCs Used .....	4
E-3	CRC Usage Specifications .....	4
E-4	CRC Example Code.....	5



## E-1 Introduction

This appendix presents the details of the safety related CRCs used in the safety protocol.

**Table E-1.1 Properties of 8 Bit CRC (CRC-S1)**

	CRC-S1	CRC-S2	CRC-S3	CRC-S4	CRC-S5
Width	8	8	16	32	24
Poly	0x37	0x3b	0x080F	0xEDB88320	0x5D6DCB
Init <sup>1</sup>	Variable	Variable	Variable	Variable	Variable
RefIn	False	False	False	False	False
RefOut	False	False	False	False	False
XorOut	0x00	0x00	0x0000	0x00000000	0x00FFFFFF
Check <sup>2</sup>	0x4C	0xBF	0x9516	0x340BC6D9	

- 1 The initial values used for the runtime protocol are in Volume 5 chapter 2 for CRC-S1, CRC-S2, CRC-S3 and CRC-S5. The initial values used for CRC-S4 are in table E-3.1
- 2 To generate the correct check values use 0xFF as initial values for CRC-S1 and CRC-S2, 0xFFFF for CRC-S3, 0xFFFFFFFF for CRC-S4 and 0x00FFFFFF for CRC-S5 .

Where:

**WIDTH:** This is the width of the algorithm expressed in bits. This is one less than the width of the Poly.

**POLY:** This parameter is the poly. This is a binary value which is specified as a hexadecimal number.

**INIT:** This parameter specifies the initial value of the register when the algorithm starts. This is the value that is to be assigned to the register in the direct table algorithm. In the table algorithm, we may think of the register always commencing with the value zero, and this value being XORed into the register after the N'th bit iteration

**REFIN:** This is a boolean parameter. If it is FALSE, input bytes are processed with bit 7 being treated as the most significant bit (MSB) and bit 0 being treated as the least significant bit. If this parameter is TRUE, each byte is reflected before being processed.

**REFOUT:** This is a boolean parameter. If it is set to FALSE, the final value in the register is fed into the XOROUT stage directly, otherwise, if this parameter is TRUE, the final register value is reflected first.

**XOROUT:** This is a W-bit value that should be specified as a hexadecimal number. It is XORed to the final register value (after the REFOUT) stage before the value is returned as the official checksum.

**CHECK:** This field is not strictly part of the definition, and, in the event of an inconsistency between this field and the other field, the other fields take precedence. This field is a check value that can be used as a weak validator of implementations of the algorithm. The field contains the checksum obtained when the ASCII string "123456789" is fed through the specified algorithm (i.e. 313233... (hexadecimal)).



## E-2 Native CRCs Used

It is important that the Native CRCs be of a different form and be calculated differently than the Safety CRCs. The following table shows the CRCs used in the networks that will support the safety protocol. The native CRCs are used for diagnostics. The native CRCs will not be considered as part of the safety function of the protocol.

**Table E-2.1 Native Generator Polynomials**

Network	Native Generator Polynomial
DeviceNet	$g(x) = x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$
EtherNet/IP SERCOS III	$g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
ControlNet	$g(x) = x^{16} + x^{12} + x^5 + 1$

All of the networks that support the safety protocol calculate the native CRC in the network interface chips or ASICs. The Safety CRCs will be calculated using different circuits, algorithms or software.

## E-3 CRC Usage Specifications

The I/O messaging CRC usage is defined in Chapter 2 using CRC-S1, CRC-S2, CRC-S3, and CRC-S5 but there are a number of other CRCs defined in the safety protocol for connection establishment, configuration, and the Safety extension to the EDS file definition. Table E-3.1 summarizes which algorithm is used and the starting seed value for each use.

**Table E-3.1 CRC Usage for Connection and Configuration**

CRC Name	Algorithm Used	Starting Seed Value	Spec Reference	Description
SCCRC	CRC-S4 (refer to example code)	0xFFFFFFFF	Section 2 – 10.3.2	Safety Configuration CRC
CPCRC	CRC-S4 (refer to example code)	0xFFFFFFFF	Section 2 – 10.3.6	Connection Parameter CRC
Password CRC	CRC-S4 (refer to example code)	0xFDFDFDFD	Section 7- 1.1.4	Used to encrypt passwords before transmission
EDS File CRC	CRC-S4 (refer to example code)	0xFDFDFDFD	Section 7 - 2.2.1.1	Used to provide integrity over EDS files used by safety devices



**E-4 CRC Example Code**

```

#include "stdio.h"      //for printf
#include "memory.h"     //for memcmp

#define CRCS1_POLYNOMIAL 0x37
#define CRCS2_POLYNOMIAL 0x3b
#define CRCS3_POLYNOMIAL 0x080F
#define CRC32_POLYNOMIAL 0xedb88320 // not 0x04c11db7
#define CRCS5_POLYNOMIAL 0x5d6dcb

// CRC32 (CRC-S4)
// polynomial 0xedb88320 - standard CRC-32 bit reversed from the regular
// published polynomial of 0x04c11db7. This table should be used to
// replicate standard Ethernet CRC hardware algorithm. Thus the software
// implementation of the Ethernet CRC should produce the same result as the
// hardware. This table was generated via right shifting the LSB of each
// byte
const unsigned long CRC32Table[256] =
{
0x00000000, 0x77073096, 0xEE0E612C, 0x990951BA,
0x076DC419, 0x706AF48F, 0xE963A535, 0x9E6495A3,
0x0EDB8832, 0x79DCB8A4, 0xE0D5E91E, 0x97D2D988,
0x09B64C2B, 0x7EB17CBD, 0xE7B82D07, 0x90BF1D91,
0x1DB71064, 0x6AB020F2, 0xF3B97148, 0x84BE41DE,
0x1ADAD47D, 0x6DDDE4EB, 0xF4D4B551, 0x83D385C7,
0x136C9856, 0x646BA8C0, 0xFD62F97A, 0x8A65C9EC,
0x14015C4F, 0x63066CD9, 0xFA0F3D63, 0x8D080DF5,
0x3B6E20C8, 0x4C69105E, 0xD56041E4, 0xA2677172,
0x3C03E4D1, 0x4B04D447, 0xD20D85FD, 0xA50AB56B,
0x35B5A8FA, 0x42B2986C, 0xDBBBC9D6, 0xACBCF940,
0x32D86CE3, 0x45DF5C75, 0xDCD60DCF, 0xABD13D59,
0x26D930AC, 0x51DE003A, 0xC8D75180, 0xBFDD06116,
0x21B4F4B5, 0x56B3C423, 0xCFBA9599, 0xB8BDA50F,
0x2802B89E, 0x5F058808, 0xC60CD9B2, 0xB10BE924,
0x2F6F7C87, 0x58684C11, 0xC1611DAB, 0xB6662D3D,
0x76DC4190, 0x01DB7106, 0x98D220BC, 0xEFD5102A,
0x71B18589, 0x06B6B51F, 0x9FBBF4A5, 0xEB8BDA33,
0x7807C9A2, 0x0F00F934, 0x9609A88E, 0xE10E9818,
0x76A5B44B, 0x05A6644D, 0x9E6495A3, 0xE963A535,
0x6B6B51F4, 0x1C6C6162, 0x856530D8, 0xF262004E,
0x6C0695ED, 0x1B01A57B, 0x8208F4C1, 0xF50FC457,
0x65B0D9C6, 0x12B7E950, 0x8BBEB8EA, 0xFCB9887C,
0x62DD1DDF, 0x15DA2D49, 0x8CD37CF3, 0xFBD44C65,
0x4DB26158, 0x3AB551CE, 0xA3BC0074, 0xD4BB30E2,
0x4ADFA541, 0x3DD895D7, 0xA4D1C46D, 0xD3D6F4FB,
0x4369E96A, 0x346ED9FC, 0xAD678846, 0xDA60B8D0,
0x44042D73, 0x33031DE5, 0xAA0A4C5F, 0xDD0D7CC9,
0x5005713C, 0x270241AA, 0xBE0B1010, 0xC90C2086,
0x5768B525, 0x206F85B3, 0xB966D409, 0xCE61E49F,
0x5EDEF90E, 0x29D9C998, 0xB0D09822, 0xC7D7A8B4,
0x59B33D17, 0x2EB40D81, 0xB7BD5C3B, 0xC0BA6CAD,
0xEDB88320, 0x9ABFB3B6, 0x03B6E20C, 0x74B1D29A,
0xEAD54739, 0x9DD277AF, 0x04DB2615, 0x73DC1683,
0xE3630B12, 0x94643B84, 0x0D6D6A3E, 0x7A6A5AA8,
0xE40ECF0B, 0x9309FF9D, 0x0A00AE27, 0x7D079EB1,
0xF00F9344, 0x8708A3D2, 0x1E01F268, 0x6906C2FE,
0xF762575D, 0x806567CB, 0x196C3671, 0x6E6B06E7,
0xFED41B76, 0x89D32BE0, 0x10DA7A5A, 0x67DD4ACC,
0xF9B9DF6F, 0x8EBEEFF9, 0x17B7BE43, 0x60B08ED5,
0xD6D6A3E8, 0xA1D1937E, 0x38D8C2C4, 0x4FDFF252,
0xD1BB67F1, 0xA6BC5767, 0x3FB506DD, 0x48B2364B,
0xD80DBDA, 0xAF0A1B4C, 0x36034AF6, 0x41047A60,
0xDF60EFC3, 0xA867DF55, 0x316E8EEF, 0x4669BE79,
0xCB61B38C, 0xBC66831A, 0x256FD2A0, 0x5268E236,
0xCC0C7795, 0xBB0B4703, 0x220216B9, 0x5505262F,
0xC5BA3BBE, 0xB2BD0B28, 0x2BB45A92, 0x5CB36A04,
0xC2D7FFA7, 0xB5D0CF31, 0x2CD99E8B, 0x5BDEAE1D,
0x9B64C2B0, 0xEC63F226, 0x756AA39C, 0x026D930A,
0x9C0906A9, 0xEB0E363F, 0x72076785, 0x05005713,

```



```

0x95BF4A82, 0xE2B87A14, 0x7BB12BAE, 0x0CB61B38,
0x92D28E9B, 0xE5D5BE0D, 0x7CDCEFB7, 0x0BDBDF21,
0x86D3D2D4, 0xF1D4E242, 0x68DDB3F8, 0x1FDA836E,
0x81BE16CD, 0xF6B9265B, 0x6FB077E1, 0x18B74777,
0x88085AE6, 0xFF0F6A70, 0x66063BCA, 0x11010B5C,
0x8F659EFF, 0xF862AE69, 0x616BFFD3, 0x166CCF45,
0xA00AE278, 0xD70DD2EE, 0x4E048354, 0x3903B3C2,
0xA7672661, 0xD06016F7, 0x4969474D, 0x3E6E77DB,
0xAED16A4A, 0xD9D65ADC, 0x40DF0B66, 0x37D83BF0,
0xA9BCAE53, 0xDEBB9EC5, 0x47B2CF7F, 0x30B5FFE9,
0xBDBDF21C, 0xCABAC28A, 0x53B39330, 0x24B4A3A6,
0xBAD03605, 0xCDD70693, 0x54DE5729, 0x23D967BF,
0xB3667A2E, 0xC4614AB8, 0x5D681B02, 0x2A6F2B94,
0xB40BBE37, 0xC30C8EA1, 0x5A05DF1B, 0x2D02EF8D
};

//CRCS3 POLYNOMIAL = 0x080F
//This table was generated via left shifting the MSB of each byte
const unsigned short CRC16Table[256] =
{
0x0000, 0x080F, 0x101E, 0x1811, 0x203C, 0x2833, 0x3022, 0x382D,
0x4078, 0x4877, 0x5066, 0x5869, 0x6044, 0x684B, 0x705A, 0x7855,
0x80F0, 0x88FF, 0x90EE, 0x98E1, 0xA0CC, 0xA8C3, 0xB0D2, 0xB8DD,
0xC088, 0xC887, 0xD096, 0xD899, 0xE0B4, 0xE8BB, 0xF0AA, 0xF8A5,
0x09EF, 0x01E0, 0x19F1, 0x11FE, 0x29D3, 0x21DC, 0x39CD, 0x31C2,
0x4997, 0x4198, 0x5989, 0x5186, 0x69AB, 0x61A4, 0x79B5, 0x71BA,
0x891F, 0x8110, 0x9901, 0x910E, 0xA923, 0xA12C, 0xB93D, 0xB132,
0xC967, 0xC168, 0xD979, 0xD176, 0xE95B, 0xE154, 0xF945, 0xF14A,
0x13DE, 0x1BD1, 0x03C0, 0x0BCF, 0x33E2, 0x3BED, 0x23FC, 0x2BF3,
0x53A6, 0x5BA9, 0x43B8, 0x4BB7, 0x739A, 0x7B95, 0x6384, 0x6B8B,
0x932E, 0x9B21, 0x8330, 0x833F, 0xB312, 0xBB1D, 0xA30C, 0xAB03,
0xD356, 0xDB59, 0xC348, 0xCB47, 0xF36A, 0xFB65, 0xE374, 0xEB7B,
0x1A31, 0x123E, 0x0A2F, 0x0220, 0x3A0D, 0x3202, 0x2A13, 0x221C,
0x5A49, 0x5246, 0x4A57, 0x4258, 0x7A75, 0x727A, 0x6A6B, 0x6264,
0x9AC1, 0x92CE, 0x8ADF, 0x82D0, 0xBAFD, 0xB2F2, 0xAAE3, 0xA2EC,
0xDAB9, 0xD2B6, 0xCAA7, 0xC2A8, 0xFA85, 0xF28A, 0xEA9B, 0xE294,
0x27BC, 0x2FB3, 0x37A2, 0x3FAD, 0x0780, 0x0F8F, 0x179E, 0x1F91,
0x67C4, 0x6FCB, 0x77DA, 0x7FD5, 0x47F8, 0x4FF7, 0x57E6, 0x5FE9,
0xA74C, 0xAF43, 0xB752, 0xBF5D, 0x8770, 0x8F7F, 0x976E, 0x9F61,
0xE734, 0xEF3B, 0xF72A, 0xFF25, 0xC708, 0xCF07, 0xD716, 0xDF19,
0x2E53, 0x265C, 0x3E4D, 0x3642, 0x0E6F, 0x0660, 0x1E71, 0x167E,
0x6E2B, 0x6624, 0x7E35, 0x763A, 0x4E17, 0x4618, 0x5E09, 0x5606,
0xAEA3, 0xA6AC, 0xBEBD, 0xB6B2, 0x8E9F, 0x8690, 0x9E81, 0x968E,
0xEEDB, 0xEDD4, 0xFEC5, 0xF6CA, 0xCEE7, 0xC6E8, 0xDE99, 0xD6F6,
0x3462, 0x3C6D, 0x247C, 0x2C73, 0x145E, 0x1C51, 0x0440, 0x0C4F,
0x741A, 0x7C15, 0x6404, 0x6C0B, 0x5426, 0x5C29, 0x4438, 0x4C37,
0xB492, 0xBC9D, 0xA48C, 0xAC83, 0x94AE, 0x9CA1, 0x84B0, 0x8CBF,
0xF4EA, 0xFCF5, 0xE4F4, 0xECFB, 0xD4D6, 0xDCD9, 0xC4C8, 0xCCC7,
0x3D8D, 0x3582, 0x2D93, 0x259C, 0x1DB1, 0x15BE, 0x0DAF, 0x05A0,
0x7DF5, 0x75FA, 0x6DEB, 0x65E4, 0x5DC9, 0x55C6, 0x4DD7, 0x45D8,
0xBD7D, 0xB572, 0xAD63, 0xA56C, 0x9D41, 0x954E, 0x8D5F, 0x8550,
0xFD05, 0xF50A, 0xED1B, 0xE514, 0xDD39, 0xD536, 0xCD27, 0xC528
};

//CRCS1 POLYNOMIAL=0x37
//This table was generated via left shifting the MSB of each byte
const unsigned char CRC8Table37[256] =
{
0x00, 0x37, 0x6e, 0x59, 0xdc, 0xeb, 0xb2, 0x85,
0x8f, 0xb8, 0xe1, 0xd6, 0x53, 0x64, 0x3d, 0x0a,
0x29, 0x1e, 0x47, 0x70, 0xf5, 0xc2, 0x9b, 0xac,
0xa6, 0x91, 0xc8, 0xff, 0x7a, 0x4d, 0x14, 0x23,
0x52, 0x65, 0x3c, 0x0b, 0x8e, 0xb9, 0xe0, 0xd7,
0xdd, 0xea, 0xb3, 0x84, 0x01, 0x36, 0x6f, 0x58,
0x7b, 0x4c, 0x15, 0x22, 0xa7, 0x90, 0xc9, 0xfe,
0xf4, 0xc3, 0x9a, 0xad, 0x28, 0x1f, 0x46, 0x71,
0xa4, 0x93, 0xca, 0xfd, 0x78, 0x4f, 0x16, 0x21,
0x2b, 0x1c, 0x45, 0x72, 0xf7, 0xc0, 0x99, 0xae,
0x8d, 0xba, 0xe3, 0xd4, 0x51, 0x66, 0x3f, 0x08,
0x02, 0x35, 0x6c, 0x5b, 0xde, 0xe9, 0xb0, 0x87,
0xf6, 0xc1, 0x98, 0xaf, 0x2a, 0x1d, 0x44, 0x73,
0x79, 0x4e, 0x17, 0x20, 0xa5, 0x92, 0xcb, 0xfc,

```



```

0xdf, 0xe8, 0xb1, 0x86, 0x03, 0x34, 0x6d, 0x5a,
0x50, 0x67, 0x3e, 0x09, 0x8c, 0xbb, 0xe2, 0xd5,
0x7f, 0x48, 0x11, 0x26, 0xa3, 0x94, 0xcd, 0xfa,
0xf0, 0xc7, 0x9e, 0xa9, 0x2c, 0x1b, 0x42, 0x75,
0x56, 0x61, 0x38, 0x0f, 0x8a, 0xbd, 0xe4, 0xd3,
0xd9, 0xee, 0xb7, 0x80, 0x05, 0x32, 0x6b, 0x5c,
0x2d, 0x1a, 0x43, 0x74, 0xf1, 0xc6, 0x9f, 0xa8,
0xa2, 0x95, 0xcc, 0xfb, 0x7e, 0x49, 0x10, 0x27,
0x04, 0x33, 0x6a, 0x5d, 0xd8, 0xef, 0xb6, 0x81,
0x8b, 0xbc, 0xe5, 0xd2, 0x57, 0x60, 0x39, 0x0e,
0xdb, 0xec, 0xb5, 0x82, 0x07, 0x30, 0x69, 0x5e,
0x54, 0x63, 0x3a, 0x0d, 0x88, 0xbf, 0xe6, 0xd1,
0xf2, 0xc5, 0x9c, 0xab, 0x2e, 0x19, 0x40, 0x77,
0x7d, 0x4a, 0x13, 0x24, 0xa1, 0x96, 0xcf, 0xf8,
0x89, 0xbe, 0xe7, 0xd0, 0x55, 0x62, 0x3b, 0x0c,
0x06, 0x31, 0x68, 0x5f, 0xda, 0xed, 0xb4, 0x83,
0xa0, 0x97, 0xce, 0xf9, 0x7c, 0x4b, 0x12, 0x25,
0x2f, 0x18, 0x41, 0x76, 0xf3, 0xc4, 0x9d, 0xaa,
};

//CRCS2_POLYNOMIAL = 0x3b
//This table was generated via left shifting the MSB of each byte
const unsigned char CRC8Table3b[256] =
{
0x00, 0x3b, 0x76, 0x4d, 0xec, 0xd7, 0x9a, 0xa1,
0xe3, 0xd8, 0x95, 0xae, 0x0f, 0x34, 0x79, 0x42,
0xfd, 0xc6, 0x8b, 0xb0, 0x11, 0x2a, 0x67, 0x5c,
0x1e, 0x25, 0x68, 0x53, 0xf2, 0xc9, 0x84, 0xbf,
0xc1, 0xfa, 0xb7, 0x8c, 0x2d, 0x16, 0x5b, 0x60,
0x22, 0x19, 0x54, 0x6f, 0xce, 0xf5, 0xb8, 0x83,
0x3c, 0x07, 0x4a, 0x71, 0xd0, 0xeb, 0xa6, 0x9d,
0xdf, 0xe4, 0xa9, 0x92, 0x33, 0x08, 0x45, 0x7e,
0xb9, 0x82, 0xcf, 0xf4, 0x55, 0x6e, 0x23, 0x18,
0x5a, 0x61, 0x2c, 0x17, 0xb6, 0x8d, 0xc0, 0xfb,
0x44, 0x7f, 0x32, 0x09, 0xa8, 0x93, 0xde, 0xe5,
0xa7, 0x9c, 0xd1, 0xea, 0x4b, 0x70, 0x3d, 0x06,
0x78, 0x43, 0x0e, 0x35, 0x94, 0xaf, 0xe2, 0xd9,
0x9b, 0xa0, 0xed, 0xd6, 0x77, 0x4c, 0x01, 0x3a,
0x85, 0xbe, 0xf3, 0xc8, 0x69, 0x52, 0x1f, 0x24,
0x66, 0x5d, 0x10, 0x2b, 0x8a, 0xb1, 0xfc, 0xc7,
0x49, 0x72, 0x3f, 0x04, 0xa5, 0x9e, 0xd3, 0xe8,
0xaa, 0x91, 0xdc, 0xe7, 0x46, 0x7d, 0x30, 0x0b,
0xb4, 0x8f, 0xc2, 0xf9, 0x58, 0x63, 0x2e, 0x15,
0x57, 0x6c, 0x21, 0x1a, 0xbb, 0x80, 0xcd, 0xf6,
0x88, 0xb3, 0xfe, 0xc5, 0x64, 0x5f, 0x12, 0x29,
0x6b, 0x50, 0x1d, 0x26, 0x87, 0xbc, 0xf1, 0xca,
0x75, 0x4e, 0x03, 0x38, 0x99, 0xa2, 0xef, 0xd4,
0x96, 0xad, 0xe0, 0xdb, 0x7a, 0x41, 0x0c, 0x37,
0xf0, 0xcb, 0x86, 0xbd, 0x1c, 0x27, 0x6a, 0x51,
0x13, 0x28, 0x65, 0x5e, 0xff, 0xc4, 0x89, 0xb2,
0x0d, 0x36, 0x7b, 0x40, 0xe1, 0xda, 0x97, 0xac,
0xee, 0xd5, 0x98, 0xa3, 0x02, 0x39, 0x74, 0x4f,
0x31, 0x0a, 0x47, 0x7c, 0xdd, 0xe6, 0xab, 0x90,
0xd2, 0xe9, 0xa4, 0x9f, 0x3e, 0x05, 0x48, 0x73,
0xcc, 0xf7, 0xba, 0x81, 0x20, 0x1b, 0x56, 0x6d,
0x2f, 0x14, 0x59, 0x62, 0xc3, 0xf8, 0xb5, 0x8e,
};

// polynomial 0x5D6DCB
// This table was generated via left shifting the MSB of each byte and appending an
// additional byte to form a long word value. The MSB of the table value will
// always be zero
const unsigned long CRCS5Table[256] =
{
0x00000000, 0x005d6dcb, 0x00badb96, 0x00e7b65d,
0x0028dae7, 0x0075b72c, 0x00920171, 0x00cfc6ba,
0x0051b5ce, 0x000cd805, 0x00eb6e58, 0x00b60393,
0x00796f29, 0x002402e2, 0x00c3b4bf, 0x009ed974,
0x00a36b9c, 0x00fe0657, 0x0019b00a, 0x0044ddc1,
0x008bb17b, 0x00d6dcb0, 0x00316aed, 0x006c0726,
0x00f2de52, 0x00afb399, 0x004805c4, 0x0015680f,
0x00da04b5, 0x0087697e, 0x0060df23, 0x003db2e8,

```



```

0x001bbaf3, 0x0046d738, 0x00a16165, 0x00fc0cae,
0x00336014, 0x006e0ddf, 0x0089bb82, 0x00d4d649,
0x004a0f3d, 0x001762f6, 0x00f0d4ab, 0x00adb960,
0x0062d5da, 0x003fb811, 0x00d80e4c, 0x00856387,
0x00b8d16f, 0x00e5bca4, 0x00020af9, 0x005f6732,
0x00900b88, 0x00cd6643, 0x002ad01e, 0x0077bdd5,
0x00e964a1, 0x00b4096a, 0x0053bf37, 0x000ed2fc,
0x00c1be46, 0x009cd38d, 0x007b65d0, 0x0026081b,
0x003775e6, 0x006a182d, 0x008dae70, 0x00d0c3bb,
0x001faf01, 0x0042c2ca, 0x00a57497, 0x00f8195c,
0x0066c028, 0x003bade3, 0x00dc1bbe, 0x00817675,
0x004e1acf, 0x00137704, 0x00f4c159, 0x00a9ac92,
0x00941e7a, 0x00c973b1, 0x002ec5ec, 0x0073a827,
0x00bcc49d, 0x00e1a956, 0x00061f0b, 0x005b72c0,
0x00c5abb4, 0x0098c67f, 0x007f7022, 0x00221de9,
0x00ed7153, 0x00b01c98, 0x0057aac5, 0x000ac70e,
0x002ccf15, 0x0071a2de, 0x00961483, 0x00cb7948,
0x000415f2, 0x00597839, 0x00bece64, 0x00e3a3af,
0x007d7adb, 0x00201710, 0x00c7a14d, 0x009acc86,
0x0055a03c, 0x0008cdf7, 0x00ef7baa, 0x00b21661,
0x008fa489, 0x00d2c942, 0x00357f1f, 0x006812d4,
0x00a77e6e, 0x00fa13a5, 0x001da5f8, 0x0040c833,
0x00de1147, 0x00837c8c, 0x0064cad1, 0x0039a71a,
0x00f6cba0, 0x00aba66b, 0x004c1036, 0x00117dfd,
0x006eebcc, 0x00338607, 0x00d4305a, 0x00895d91,
0x0046312b, 0x001b5ce0, 0x00fceabd, 0x00a18776,
0x003f5e02, 0x006233c9, 0x00858594, 0x00d8e85f,
0x001784e5, 0x004ae92e, 0x00ad5f73, 0x00f032b8,
0x00cd8050, 0x0090ed9b, 0x00775bc6, 0x002a360d,
0x00e55ab7, 0x00b8377c, 0x005f8121, 0x0002ecea,
0x009c359e, 0x00c15855, 0x0026ee08, 0x007b83c3,
0x00b4ef79, 0x00e982b2, 0x000e34ef, 0x00535924,
0x0075513f, 0x00283cf4, 0x00cf8aa9, 0x0092e762,
0x005d8bd8, 0x0000e613, 0x00e7504e, 0x00ba3d85,
0x0024e4f1, 0x0079893a, 0x009e3f67, 0x00c352ac,
0x000c3e16, 0x005153dd, 0x00b6e580, 0x00eb884b,
0x00d63aa3, 0x008b5768, 0x006ce135, 0x00318cfe,
0x00fee044, 0x00a38d8f, 0x00443bd2, 0x00195619,
0x00878f6d, 0x00dae2a6, 0x003d5afb, 0x00603930,
0x00af558a, 0x00f23841, 0x00158e1c, 0x0048e3d7,
0x00599e2a, 0x0004f3e1, 0x00e345bc, 0x00be2877,
0x007144cd, 0x002c2906, 0x00cb9f5b, 0x0096f290,
0x00082be4, 0x0055462f, 0x00b2f072, 0x00ef9db9,
0x0020f103, 0x007d9cc8, 0x009a2a95, 0x00c7475e,
0x00faf5b6, 0x00a7987d, 0x00402e20, 0x001d43eb,
0x00d22f51, 0x008f429a, 0x0068f4c7, 0x0035990c,
0x00ab4078, 0x00f62db3, 0x00119bee, 0x004cf625,
0x00839a9f, 0x00def754, 0x00394109, 0x00642cc2,
0x004224d9, 0x001f4912, 0x00f8ff4f, 0x00a59284,
0x006afe3e, 0x003793f5, 0x00d025a8, 0x008d4863,
0x00139117, 0x004efcdc, 0x00a94a81, 0x00f4274a,
0x003b4bf0, 0x0066263b, 0x00819066, 0x00dcfdad,
0x00e14f45, 0x00bc228e, 0x005b94d3, 0x0006f918,
0x00c995a2, 0x0094f869, 0x00734e34, 0x002e23ff,
0x00b0fa8b, 0x00ed9740, 0x000a211d, 0x00574cd6,
0x0098206c, 0x00c54da7, 0x0022fbfa, 0x007f9631
};

/** GetCRC32Ethernet

\par Purpose:
CRC32 calculation routine for polynomial 0xedb88320 which is bit reversed
from the regular polynomial of 0x04c11db7 thus the (crc >> 8) below
is right shifted.

\note
Doing incremental crc calculation is done with providing the result
from the previous block with preset.

\note
Restrictions:
    len > 0

```



```
pStart points to valid memory with at least len bytes in size

\param Input: \b pStart      Starting address to compute CRC over
\param Input: \b len        Number of bytes
\param Input: \b preset     Preset value for CRC

*/
unsigned long GetCRC32Ethernet(const void *pStart, int len, unsigned long preset)
{
    unsigned long crc = preset;
    unsigned char *buf = (unsigned char *)pStart;

    while (len-- > 0)
    {
        unsigned char data8 = *buf++;
        crc = CRC32Table[(crc ^ data8) & 0xff] ^ (crc >> 8);
    }
    return crc;
} //GetCRC32Ethernet

/** ComputeCRCS1

\param Purpose:
CRC8 calculation routine for polynomial 0x37.
Note that the CRC8Table37 was computed via left shift operations.

\note
Doing incremental crc calculation is done with providing the result
from the previous block with preset.

\note
Restrictions:
    len > 0
    pStart points to valid memory with at least len bytes in size

\param Scope: public

\param Input: \b pStart      Starting address to compute CRC over
\param Input: \b len        Number of bytes
\param Input: \b preset     Preset value for CRC

\param Edit History:

*/
unsigned char ComputeCRCS1(const void *pStart, int len, unsigned char preset)
{
    unsigned char crc = preset;
    unsigned char *buf = (unsigned char *)pStart;

    while (len-- > 0)
    {
        crc = CRC8Table37[(crc ^ *buf++)];
    }
    return (crc);
} //ComputeCRCS1

/** ComputeCRCS2

\param Purpose:
CRC8 calculation routine for polynomial 0x3b.
Note that the CRC8Table3b was computed via left shift operations.

\note
Doing incremental crc calculation is done with providing the result
from the previous block with preset.

\note
Restrictions:
    len > 0
    pStart points to valid memory with at least len bytes in size
```



```
\par Scope: public

\param Input: \b pStart      Starting address to compute CRC over
\param Input: \b len        Number of bytes
\param Input: \b preset     Preset value for CRC

\par Edit History:
*/
unsigned char ComputeCRCS2(const void *pStart, int len, unsigned char preset)
{
    unsigned char crc = preset;
    unsigned char *buf = (unsigned char *)pStart;

    while (len-- > 0)
    {
        crc = CRC8Table3b[(crc ^ *buf++)];
    }
    return (crc);
} //ComputeCRCS2

/** ComputeCRCS3

\par Purpose:
CRC16 calculation routine for polynomial 0x080F.
The CRC16Table was computed via left shift operations and thus will do
a (crc << 8).

\note
Doing incremental crc calculation is done with providing the result
from the previous block with preset.

\note
Restrictions:
    len > 0
    pStart points to valid memory with at least len bytes in size

\param Input: \b pStart      Starting address to compute CRC over
\param Input: \b len        Number of bytes
\param Input: \b preset     Preset value for CRC

*/
unsigned short ComputeCRCS3(const void *pStart, int len, unsigned short preset)
{
    unsigned short crc = preset;
    unsigned char *buf = (unsigned char *)pStart;

    while (len-- > 0)
    {
        unsigned char data = *buf++;
        crc = CRC16Table[((crc >> 8) ^ data)] ^ (crc << 8);
    }
    return (crc);
} //ComputeCRCS3

/** ComputeCRCS3Ref4

\par Purpose:
Compute crc given 4 bytes of data and a preset for the crc.
This assumes the input data is stored in native (little or big)
endian format.

\par Scope: public

\param Input: \b data        Value to compute CRC on
\param Input: \b preset     Preset value for CRC

\par Edit History:
*/
unsigned short ComputeCRCS3Ref4(unsigned long data, unsigned short preset,
```



```

{
    unsigned short crc = preset;
    crc = CRC16Table[(crc >> 8) ^ (data&0xff)] ^ (crc << 8);
    crc = CRC16Table[(crc >> 8) ^ ((data>>8)&0xff)] ^ (crc << 8);
    crc = CRC16Table[(crc >> 8) ^ ((data>>16)&0xff)] ^ (crc << 8);
    crc = CRC16Table[(crc >> 8) ^ ((data>>24)&0xff)] ^ (crc << 8);
    return crc;
}

/** ComputeCRCS3Ref2

\par Purpose:
Compute crc given 2 bytes of data and a preset for the crc.
This assumes the input data is stored in native (little or big) endian format.

\par Scope: public

\param Input: \b data      Value to compute CRC on
\param Input: \b preset    Preset value for CRC

\par Edit History:
*/
unsigned short ComputeCRCS3Ref2(unsigned short data, unsigned short preset)
{
    unsigned short crc = preset;
    crc = CRC16Table[(crc >> 8) ^ (data&0xff)] ^ (crc << 8);
    crc = CRC16Table[(crc >> 8) ^ ((data>>8)&0xff)] ^ (crc << 8);
    return crc;
}

/** ComputeCRCS5

\par Purpose:
CRCS5 calculation routine for polynomial 0x5D6DCB.
The CRCS5Table was computed via left shift operations and thus will do a (crc
<< 8).
\note
Doing incremental crc calculation is done with providing the result
from the previous block with preset.

\note
Restrictions:
    len > 0
    pStart points to valid memory with at least len bytes in size.
    CRC value returned is a long with the MSB = 0

\param Input: \b pStart    Starting address to compute CRC over
\param Input: \b len       Number of bytes
\param Input: \b preset    Preset value for CRC (unsigned long)

*/
unsigned long ComputeCRCS24(const void *pStart, int len, unsigned long preset)
{
    unsigned long crc = preset;
    unsigned long temp;
    unsigned char *buf = (unsigned char *)pStart;

    while (len-- > 0)
    {
        unsigned char data = *buf++;
        // XOR data with CRC2, look up result, then XOR that with CRC;
        crc = CRCS5Table[((crc >> 16) ^ data)&0xff] ^ (crc << 8);
    }
    return (crc & 0x00ffffff);
} //ComputeCRCS5

/** ComputeCRCS3Ref1

\par Purpose:
Compute crc given 1 bytes of data and a preset for the crc.

\par Scope: public

```



```

\param Input: \b data      Value to compute CRC on
\param Input: \b preset    Preset value for CRC

\param Edit History:

*/
unsigned short ComputeCRCS3Ref1(unsigned char data, unsigned short preset)
{
    unsigned short crc = preset;
    crc = CRC16Table[(crc >> 8) ^ data] ^ (crc << 8);
    return crc;
}

//The following routines were used to generate the above tables.
//Just as a sanity check, rerun the calculations and test that the
//tables are correct
unsigned char CRC8Table37Ref[256];
unsigned char CRC8Table3bRef[256];
unsigned short CRC16TableRef[256];
unsigned long CRC32TableRef[256];
unsigned long CRCS5TableRef[256];
//left shift MSB
static void InitSafetyCRC8Table()
{
    unsigned short j;
    unsigned char i;
    unsigned char carry8=0;
    unsigned char entry8=0;

    //CRCS1 POLYNOMIAL
    for (j = 0; j < 256; j++)
    {
        entry8 = static_cast<unsigned char>(j);
        for (i = 0; i < 8; ++i)
        {
            carry8 = entry8 & 0x80;    // carry8 is the most significant bit
            entry8 = entry8 << 1;
            if (carry8)                // if carry8 is nonzero
            {
                entry8 = entry8 ^ CRCS1_POLYNOMIAL;    // XOR by CRC polynom
            }
        }
        CRC8Table37Ref[j] = entry8;
    }
    //CRCS2 POLYNOMIAL
    for (j = 0; j < 256; j++)
    {
        entry8 = static_cast<unsigned char>(j);
        for (i = 0; i < 8; ++i)
        {
            carry8 = entry8 & 0x80;    // carry8 is the most significant bit
            entry8 = entry8 << 1;
            if (carry8)                // if carry8 is nonzero
            {
                entry8 = entry8 ^ CRCS2_POLYNOMIAL;    // XOR by CRC polynom
            }
        }
        CRC8Table3bRef[j] = entry8;
    }
} // end of initSafetyCRCTable8bit

//left shift MSB
static void InitSafetyCRC16Table()
{
    unsigned short j;
    unsigned char i;
    unsigned short carry16=0;
    unsigned short entry16=0;

    for (j = 0; j < 256; j++)
    {

```



```

    entry16 = j;
    for (i = 0; i < 16; ++i)
    {
        carry16 = entry16 & 0x8000;    // carry16 is the lowest
                                      // significant bit
        entry16 = entry16 << 1;
        if (carry16)                  // if carry16 is nonzero
        {
            entry16 = entry16 ^ CRC32_POLYNOMIAL;    // XOR by CRC polynom
        }
    }
    CRC16TableRef[j] = entry16;
}
} // end of InitSafetyCRC16Table()

//right shift LSB
static void InitSafetyCRC32Table()
{
    unsigned long j;
    unsigned char i;
    unsigned long carry32=0;
    unsigned long entry32=0;

    //right shift
    for (j = 0; j < 256; j++)
    {
        entry32 = j;
        for (i = 0; i < 8; ++i)
        {
            carry32 = entry32 & 1;    // carry16 is the lowest
                                      // significant bit
            entry32 = entry32 >> 1;
            if (carry32)              // if carry16 is nonzero
            {
                entry32 = entry32 ^ CRC32_POLYNOMIAL;    // XOR by
                                                         // CRC polynom
            }
        }
        CRC32TableRef[j] = entry32;
    }
} // end of InitSafetyCRC32Table()

//left shift MSB
static void InitSafetyCRC5Table()
{
    unsigned short j;
    unsigned char i;
    unsigned long carry24 0;
    unsigned long entry24=0;
    unsigned long poly = unsigned long(CRC5_POLYNOMIAL);

    for (j = 0; j < 256; j++)
    {
        entry24 = (j<<16);
        for (i = 0; i < 8; ++i)
        {
            carry24 = entry24 & 0x800000;    // carry24 is the most significant
bit
            entry24 = entry24 << 1;
            if (carry24)                  // if carry24 is nonzero
            {
                entry24 = entry24 ^ poly;    // XOR by CRC polynom
            }
        }
        CRC5TableRef[j] = entry24 & 0x00ffffff; // Upper byte is don't care
    }
} // end of InitSafetyCRC5Table()

//This is the reference model that the table driven routine must match
unsigned char Crc8ComputeSlow(const void* buffer, unsigned int count, unsigned

```



```

char poly, unsigned char preset)
{
    const unsigned char* ptr = (const unsigned char *) buffer;
    unsigned char crc8=preset;
    unsigned char value=0;

    while (count--)
    {
        value = *ptr++;
        crc8 ^= value;
        for (int i = 0; i < 8; i++)
        {
            if (crc8 & 0x80)
            {
                crc8 = (crc8 << 1) ^ poly;
            }
            else
            {
                crc8 <<= 1;
            }
        }
    }
    return crc8;
} //Crc8ComputeSlow

//This is the reference model that the table driven routine must match
unsigned short Crc16ComputeSlow(const void* buffer, unsigned int count, unsigned
short poly, unsigned short preset)
{
    const unsigned char* ptr = (const unsigned char *) buffer;
    unsigned short crc16=preset;
    unsigned short value=0;

    while (count--)
    {
        value = *ptr++;
        crc16 ^= (value << 8);
        for (int i = 0; i < 8; i++)
        {
            if (crc16 & 0x8000)
            {
                crc16 = (crc16 << 1) ^ poly;
            }
            else
            {
                crc16 <<= 1;
            }
        }
    }
    return crc16;
} //Crc16ComputeSlow

//This is the reference model that the table driven routine must match
unsigned long Crc32ComputeSlow(const void* buffer, unsigned int count, unsigned
long poly, unsigned long preset)
{
    const unsigned char* ptr = (const unsigned char *) buffer;
    unsigned long crc32=preset;
    unsigned long value=0;

    while (count--)
    {
        value = *ptr++;
        crc32 ^= (value);
        for (int i = 0; i < 8; i++)
        {
            if (crc32 & 1)
            {
                crc32 = (crc32 >> 1) ^ poly;
            }
            else

```



```

        {
            crc32 >>= 1;
        }
    }
    return crc32;
} //Crc32ComputeSlow

//This is the reference model that the table driven routine must match
unsigned long CRCS5ComputeSlow(const void* buffer, unsigned int count, unsigned
long poly, unsigned long preset)
{
    const unsigned char* ptr = (const unsigned char *) buffer;
    unsigned long CRCS5 = preset;
    unsigned long value=0;

    while (count--)
    {
        value = *ptr++;
        CRCS5 ^= (value << 16);
        for (int i = 0; i < 8; i++)
        {
            if (CRCS5 & 0x8000000)
            {
                CRCS5 = (CRCS5 << 1) ^ poly;
            }
            else
            {
                CRCS5 << 1;
            }
        }
    }
    return (CRCS5 & 0x00ffffff);
} //CRCS5ComputeSlow

//define a structure such that it is 9 bytes long in packed format
//so as to equal the crc test string length (in DoCrcTest below)
struct SomeStruct
{
    unsigned char  aByte1;
    unsigned char  aByte2;
    unsigned char  aByte3;
    unsigned short aShort;
    unsigned long  aLong;
};

int DoCrcTest()
{
    const char *string = "123456789";
    int len = 9; //length of "123456789"
    unsigned long  crc32;
    unsigned short crc16;
    unsigned char  crc8;
    SomeStruct aStruct;

    //fill aStruct with equivalent to "123456789" in
    //little endian memory order
    aStruct.aByte1 = 0x31;
    aStruct.aByte2 = 0x32;
    aStruct.aByte3 = 0x33;
    aStruct.aShort = 0x3534;
    aStruct.aLong = 0x39383736;
    int crcMatch = 0;

    //the slow methods serve as the reference implementation
    //all other optimizations (ie table driven) must match these
    crc8 = Crc8ComputeSlow(string, len, CRCS1 POLYNOMIAL, 0xff);
    if (crc8 != 0x4C)
    {

```



```

        crcMatch |= 1;
    }
    crc8 = Crc8ComputeSlow(string, len, CRCS2_POLYNOMIAL, 0xff);
    if (crc8 != 0xBF)
    {
        crcMatch |= 2;
    }
    crc16 = Crc16ComputeSlow(string, len, CRCS3_POLYNOMIAL, 0xffff);
    if (crc16 != 0x9516)
    {
        crcMatch |= 4;
    }

    crc32 = Crc32ComputeSlow(string, len, CRC32_POLYNOMIAL, 0xffffffff);
    if (crc32 != 0x340bc6d9)
    {
        crcMatch |= 8;
    }

    crc8 = ComputeCRCS1(string, len, 0xff);
    if (crc8 != 0x4C)
    {
        crcMatch |= 0x10;
    }
    crc8 = ComputeCRCS2(string, len, 0xff);
    if (crc8 != 0xBF)
    {
        crcMatch |= 0x20;
    }
    crc16 = ComputeCRCS3(string, len, 0xffff);
    if (crc16 != 0x9516)
    {
        crcMatch |= 0x40;
    }

    crc32 = GetCRC32Ethernet(string, len, 0xffffffff);
    if (crc32 != 0x340bc6d9)
    {
        crcMatch |= 0x80;
    }

    //retest that the tables are correct
    InitSafetyCRC8Table();
    InitSafetyCRC16Table();
    InitSafetyCRC32Table();

    if (memcmp(&CRC8Table37Ref[0], &CRC8Table37[0], 256) != 0)
    {
        crcMatch |= 0x200;
    }
    if (memcmp(&CRC8Table3bRef[0], &CRC8Table3b[0], 256) != 0)
    {
        crcMatch |= 0x400;
    }
    if (memcmp(&CRC16TableRef[0], &CRC16Table[0], 512) != 0)
    {
        crcMatch |= 0x800;
    }
    if (memcmp(&CRC32TableRef[0], &CRC32Table[0], 1024) != 0)
    {
        crcMatch |= 0x1000;
    }

    //should work on both big and little endian machine
    crc16 = 0xffff;
    crc16 = ComputeCRCS3Ref1(aStruct.aByte1, crc16);
    crc16 = ComputeCRCS3Ref1(aStruct.aByte2, crc16);
    crc16 = ComputeCRCS3Ref1(aStruct.aByte3, crc16);
    crc16 = ComputeCRCS3Ref2(aStruct.aShort, crc16);
    crc16 = ComputeCRCS3Ref4(aStruct.aLong, crc16);
    if (crc16 != 0x9516)
    {

```



```
        crcMatch |= 0x2000;
    }

    return crcMatch;
}

int main(int argc, char* argv[])
{
    int result = DoCrcTest();
    if (result==0)
        printf("Tests passed!\n");
    else
        printf("Tests failed, result = 0x%x\n",result);
    return result;
}
```



This page is intentionally left blank



## **Volume 5: CIP Safety**

# **Appendix F: Safety Test Plan**



## Contents

F-1	Introduction.....	5
F-2	Scope.....	5
F-2.1	Functional tests .....	5
F-2.2	DUTs.....	5
F-2.2.1	Functionality .....	6
F-2.2.2	Point-to-point Test Configuration.....	6
F-2.3	System Tests .....	6
F-2.3.1	Test Conformance Strategy.....	7
F-2.4	Subsystem tests – White Box Tests .....	7
F-3	Black Box Tests .....	7
F-3.1	Safety Protocol Test Engine.....	7
F-3.2	TST1 - Standard DeviceNet Behavior .....	8
F-3.3	TST107 Standard EtherNet/IP Behavior.....	8
F-3.4	Basic Connection Establishment.....	9
F-3.4.1	Type 2 Connection Reception by Targets .....	9
F-3.4.1.1	TST2 - Positive Type 2 Connection Establishment Test .....	9
F-3.4.1.2	TST113 - Positive Type 2 Test for Extended Format connections.....	10
F-3.4.1.3	TST3 – Connection Initialization . ....	11
F-3.4.1.4	TST4 - Connection Parameters CRC Negative Test.....	13
F-3.4.1.5	TST5 - Type 2 SCID Checking Tests.....	14
F-3.4.1.6	TST6 - Electronic Key Mismatch test .....	15
F-3.4.1.7	TST7 –Target Connection Id Allocation Tests.....	16
F-3.4.1.8	TST8 - Multi-cast Producer, Consumer Number Allocation.....	17
F-3.4.1.9	TST99 – DeviceNet Base Format Multi-cast Producer Time Correction Connections.....	18
F-3.4.1.10	TST120 – DeviceNet Extended Format Multi-cast Producer Time Correction Connections and Rollover Seeding.....	18
F-3.4.2	Type 2 Connection Generation by Originators .....	19
F-3.4.2.1	TST9 - Positive Type 2 Single-Cast Connections on DeviceNet .....	19
F-3.4.2.2	TST118 - Positive Type 2 Single-Cast Connections on EtherNet/IP .....	21
F-3.4.2.3	TST10 - Positive Type 2 Multi-cast Connections on DeviceNet .....	22
F-3.4.2.4	TST119 - Positive Type 2 Multi-cast Connections on EtherNet/IP . ....	23
F-3.4.2.5	TST114- Positive Extended Format Type 2 Single-Cast Connection Origination.....	24
F-3.4.2.6	TST115- Positive Extended Format Type 2 Multi-Cast Connection Origination .....	25
F-3.4.2.7	TST100 Electronic Key Generation Test .....	26
F-3.4.3	SafetyClose Tests.....	26
F-3.4.3.1	TST101 SafetyClose Processing by Targets.....	26
F-3.5	Common Run-Time Tests .....	27
F-3.5.1	Positive Producer Tests .....	28
F-3.5.1.1	TST13 Producer CRC & PID/CID Test .....	28
F-3.5.1.2	Producer Data Message Generation.....	29
F-3.5.1.3	TST16 - Producer Packet Generation Mode Byte .....	32
F-3.5.1.4	TST17 – Base Format Producer Packet Time Stamp CRC .....	32
F-3.5.1.5	TST18 - Producer Time Correction CRC .....	33
F-3.5.1.6	TST19 - Producer Time Correction Mcast Byte & Mcast Byte 2 .....	34
F-3.5.1.7	TST111 – Extended Format Producer Time Correction Mcast Byte. ....	34
F-3.5.1.8	TST20 – Base Format Producer Single-Cast.....	35
F-3.5.1.9	TST121 – Extended Format Producer Single-Cast.....	36
F-3.5.1.10	TST21 - Producer Run/Idle Usage .....	37
F-3.5.1.11	TST22 - Producer Ping Count Usage .....	38
F-3.5.1.12	TST23 – Base Format Multi-Cast Production to a Single Consumer.....	39
F-3.5.1.13	TST122 – Extended Format Multi-Cast Production to a Single Consumer .....	40
F-3.5.1.14	TST24 - Multi-Cast Production to Multiple Consumers .....	42



F-3.5.2	Positive Consumer Tests .....	43
F-3.5.2.1	TST25 – Single-cast Consumer Time Coordination Message Generation .....	43
F-3.5.2.2	TST26 – Consumer Positive CRC/PID/CID Test.....	44
F-3.5.2.3	TST27 – Multi-cast Consumer Time Coordination Message Generation Test.....	45
F-3.6	Common Negative Consumer Tests.....	46
F-3.6.1	TST28 – Base Format Incorrect Sequence/Insertion Detection .....	46
F-3.6.2	TST112 – Extended Format Incorrect Sequence/Insertion Detection.....	47
F-3.6.3	TST29 - Message Corruption Detection .....	47
F-3.6.4	TST30 - Message Delay Detection by Consumers .....	48
F-3.7	Common Negative Producer Tests.....	49
F-3.7.1	TST31 – Base Format Producer Time Coordination Response Failure .....	49
F-3.7.2	TST116 - Extended Format Producer Time Coordination Response Failure .....	51
F-3.7.3	TST32 - Producer Time Coordination No-response ..	53
F-3.8	Single_Cast Consumer Negative Tests .....	54
F-3.8.1	TST33 – Base Format Single-Cast Consumer Test; >= 3 Bytes, Negative.....	54
F-3.8.2	TST123 Extended Format Single-Cast Consumer Test; >= 3 Bytes, Negative .....	55
F-3.8.3	TST34 – Base format Single-Cast Consumer; <= 2 Bytes, Negative ..	57
F-3.8.4	TST124 – Extended Format Single-Cast Consumer; <= 2 Bytes, Negative .....	59
F-3.8.5	TST36 - Single-Cast Consumer Communication Failure .....	60
F-3.9	Multi-Cast Connections .....	62
F-3.9.1	TST37 – Base Format Multi-Cast Consumer Negative .....	62
F-3.9.2	TST117 – Extended Format Multi-Cast Consumer Negative .....	63
F-3.9.2.1	TST38 – Base Format Multi-cast Consumer PID.....	65
F-3.9.2.2	TST125 – Extended Format Multi-cast Consumer PID/Rollover .....	66
F-3.9.2.3	TST39 - Multi-cast Consumer Communication Failure Response .....	67
F-3.9.2.4	TST40 – Base Format Multi-cast Consumer Time Correction Negative .....	68
F-3.9.2.5	TST126 – Extended Format Multi-cast Consumer Time Correction Negative .....	69
F-3.9.2.6	TST41 - Multi-cast Producer Time Correction Generation Negative.....	71
F-3.10	Device Configuration.....	72
F-3.10.1	Target Configuration .....	72
F-3.10.1.1	Configuration Ownership .....	72
F-3.10.1.2	TST42 - Configuration UNID .....	72
F-3.10.1.3	TST43 - Input Type 1 Connection Establishment ..	73
F-3.10.1.4	TST44 - Output Type 1 Connection Establishment. ....	74
F-3.10.2	Originators .....	76
F-3.10.2.1	Originators Configuring Targets .....	76
F-3.10.2.2	Connection Configuration Object Tests .....	78
F-3.11	SNCT Interface Tests.....	79
F-3.11.1	TST48 - Identity Object Tests.....	79
F-3.11.2	TST50 – MacId, SNN, and UNID Behavior Tests .....	80
F-3.11.3	TST108– IP address switches, SNN, and UNID Behavior .....	81
F-3.11.4	TST51 - Reset Switch Test .....	83
F-3.11.5	Safety Supervisor Functions .....	83
F-3.11.5.1	TST104 - Baseline Supervisor Test.....	83
F-3.11.5.2	TST52 - Propose/Apply TUNID and Reset Command Tests .....	83
F-3.11.5.3	TST53 - Configuration Lock/Unlock Test .....	91
F-3.11.5.4	TST54 - Configure_Request Test.....	93
F-3.11.5.5	TST55 - Configuration Process Test .....	94
F-3.11.5.6	TST56 - Setting Passwords Test.....	96
F-3.11.5.7	TST57 - Mode Change Test .....	97
F-3.11.6	Safety Validator Object Tests .....	97
F-3.11.6.1	TST58 - Safety Connection Diagnostics ..	97
F-3.12	DeviceNet Bridge Device Tests .....	98
F-3.12.1	TST62 - DeviceNet Bridge FW_Open to SafetyOpen Conversion .....	98
F-3.12.2	TST63 - DeviceNet Bridge Non-Dnet-to-Dnet Forwarding Test .....	98



F-3.12.3	TST64 - DeviceNet Bridge Dnet-to-nonDnet Forwarding Test.....	99
F-3.13	Standard CIP Message Interaction.....	100
F-3.13.1	Standard Messaging Rejection.....	100
F-3.13.1.1	TST65 - Standard messaging to Safety I/O.....	100
F-4	White Box Tests.....	100
F-4.1	Application Behavior.....	101
F-4.1.1	TST61 - Safety Input Application Behavior Test.....	101
F-4.1.2	TST60 - Safety Output Application Behavior Test.....	101
F-4.2	Consumer Behavior.....	102
F-4.2.1	TST66 - Consumer Initialization.....	102
F-4.2.2	TST67 - Consumer Mode Byte Processing.....	102
F-4.2.3	TST68 - Consumer Time Value Sampling.....	102
F-4.2.4	TST69 - Consumer Time Coordination Generation.....	102
F-4.2.5	TST70 - Consumer Timeout Multiplier Handling.....	103
F-4.2.6	TST71 - Multi-Cast Consumer Time Correction Handling.....	103
F-4.2.7	TST72 - Message Age Failure Indication.....	104
F-4.2.8	Link-Triggered Behavior.....	104
F-4.2.8.1	TST73 - Link-Triggered Consumer Behavior Checks.....	104
F-4.2.9	Application-Triggered Behavior.....	104
F-4.2.9.1	TST74 & TST75 - Application Triggered Consumer Behavior Checks.....	104
F-4.2.10	TST76 - Data Integrity Check Time Limit.....	105
F-4.2.11	TST77 - Safety Consumer/Application Interface.....	105
F-4.3	Producer Behavior.....	106
F-4.3.1	TST78 - Producer Initialization.....	106
F-4.3.2	TST79 - Producer Mode Byte Processing.....	106
F-4.3.3	TST80 - Producer Time Coordination Response Handling.....	106
F-4.3.4	TST85 - Producer Parameter Calculations.....	107
F-4.3.5	TST86 - Safety Producer/Application Interface.....	108
F-4.3.6	TST87 - Connection Establishment RunTime Coordination.....	108
F-4.3.7	TST127 - Extended Format Connection Establishment.....	109
F-4.4	TST88 - Safety Supervisor Behavior.....	109
F-4.5	TST 106 - Safety Supervisor Reset Password.....	109
F-4.6	TST89 - Safety Validator Behavior.....	109
F-4.7	TST90 - Corrupted Message Handling.....	110
F-4.8	TST91 - Major Fault Effect on Safety State.....	110
F-4.9	TST92 - Configuration Integrity Test.....	110
F-4.9.1	TST47 - CPCRC Handling in Originators.....	111
F-4.10	TST93 - Safety Device Hardware Validation Tests.....	111
F-4.10.1	TST103 - Switch Behavior Tests.....	111
F-4.11	TST95 - Configuration Software Tests.....	111
F-4.11.1	TST46 - Incorrect Parameter Tests for Originator Connection Configuration.....	112
F-4.12	TST96 - EDS File Requirements.....	114
F-4.13	TST97 - Safety Manual Inspection Requirement.....	114
F-4.14	TST98 - Requirements Not Tested at this time.....	115
F-4.15	TST12 - Origination of Off-link Connection Test.....	115
F-4.15.1	Test Numbers Removed.....	116
F-5	Safety Test Matrix.....	117
F-5.1	Safety Test Matrix.....	117



## **F-1 Introduction**

This document describes tests and test procedures which can be used to qualify a device that implements the Safety Protocol for DeviceNet and EtherNet/IP Safety to levels up to and including Edition 2.2. Each test lists the requirements defined in Volume 5 that would be confirmed by the test. Every requirement in Volume 5 has been mapped to a test in this document. Not all devices will implement all aspects of the protocol, so each developer will need to determine which tests apply. A function-versus-test table is provided in Section F-5 to allow implementers to determine which tests should be run for their devices. For example, some devices may not initiate connections, so they would not need to test the connection originator capability. EtherNet/IP devices may not implement some of the functions that are DeviceNet specific, so they would exclude those related tests and vice-versa.

This document will not make assumptions about how these tests will get created or how they will operate. The focus will be on “what” tests need to be performed and not “how”. However, every effort has been made to identify tests or methods that can be easily facilitated by automated test methods. The suggested test procedures described here are intentionally kept general to allow the test developer more freedom to refine the test or add more detailed steps.

Some of the tests that will be required in this document cannot be verified via black box testing. These tests will be identified as White Box Tests. It is the responsibility of the product company to document procedures for these tests during development.

This document is organized by major functionality to allow developers to quickly isolate the tests they will need to focus on for their product.

## **F-2 Scope**

This document will define tests for products that implement the safety protocol on networks DeviceNet and EtherNet/IP.

### **F-2.1 Functional tests**

The purpose of functional testing is to verify that the DUT(s) function as described in this specification.

Example: Verify proper behavior of Safety Connection Types.

### **F-2.2 DUTs**

The DUTs covered in this test document are the following:

- Any Safety Output device that utilizes the DeviceNet or EtherNet/IP Safety Protocol
- Any Safety Input device that utilizes the DeviceNet or EtherNet/IP Safety Protocol
- Any Safety Controller device that utilizes the DeviceNet or EtherNet/IP Safety Protocol



### F-2.2.1 Functionality

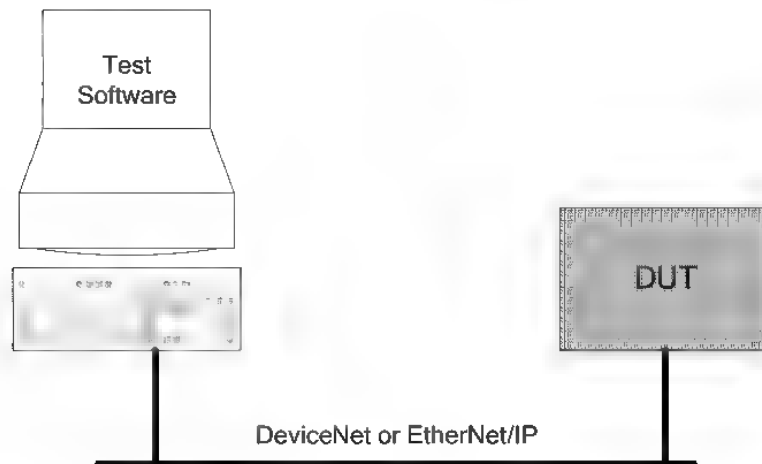
Refer to the Subsections of this document for the functionality that will be tested. Each requirement in Volume 5 has been listed under either a Black Box Test or a White Box Test to verify that functionality. It is the responsibility of either the Product Developer or the Safety Functional Test software to ensure that functionality is thoroughly tested. Test suites will be written for tests in the Black Box category. The degree to which these tests can be automated is yet to be determined since some tests require the tester to make visual confirmations.

### F-2.2.2 Point-to-point Test Configuration

The large majority of black box functional testing can be accomplished with test software running on a PC through a DeviceNet interface card or EtherNet/IP NIC card to the D.U.T. Figure below shows this basic configuration. This configuration applies whether the PC is initiating a communication transaction or D.U.T. is initiating a communication transaction. The software may change, but the basic one-on-one interface will stay the same. Tests that use this configuration will refer to it as:

Configuration: PTP

**Figure F-2.1 Safety Network Point-to-point Test Configuration**



### F-2.3 System Tests

The main purpose is to ensure that the product interfaces interact correctly with other Safety devices, such as Safety Controllers, Safety I/O, and Safety devices interacting either directly or through Bridges or Routers. Another purpose of system testing is to verify that the DUT operates or co-exists peacefully with standard products on a CIP network; that it doesn't interfere with standard devices and standard devices don't interfere with it. Most of the "Black Box" tests defined in this document can be considered System tests.

Example: Connection Establishment and Safety I/O data exchange.



### **F-2.3.1 Test Conformance Strategy**

To be able to show conformance to the protocol, all devices will execute an appropriate “set” of the system tests defined in this specification. When the test software indicates that all the required tests have passed, there is a need to have the tester make a positive confirmation via the test log files that the tests did in fact run correctly. A method will be employed in all tests that allow a tester to easily confirm that the tests were executed properly by inspecting the log files.

- Every Black Box test shall identify one or more “test milestones” that must be reached to successfully complete the test.
- Every Black Box test shall insert a special text string into the log file when the test results of each test are written to the log. This string shall contain at a minimum, the test number, milestone number, milestone name, and results indication.
- Every Black Box test shall document a pre-calculated “golden” CRC value of the “successful” milestone text strings concatenated in their anticipated order to represent a successful test execution.
- Every Black Box test shall provide a CRC calculation of the milestone text strings that are generated during the test. The final CRC value shall be inserted into the log file at the end of the test. The tester will be instructed to compare this value to the documented “golden value” to verify successful execution.
- The CRC calculation used shall be the Ethernet/IP CRC-32 algorithm defined in the Safety Functional Spec appendix C, with an initial seed value of 0xFD FD FD FD.
- Many tests require verification that Extended Format packets were dropped when errors are injected. The method used to determine this is by inspecting the Producer/Consumer Fault Count attribute of the Safety Validator object. The attribute should increment for each dropped packet on that connection.

### **F-2.4 Subsystem tests – White Box Tests**

The purpose of subsystem testing is to verify that specific functions interact correctly with other functions. Because of the nature of these tests, they need to be performed by the developer and confirmed as part of firmware qualification testing (ie. Module testing).

Example: Safety Validator interface to the device application.

## **F-3 Black Box Tests**

### **F-3.1 Safety Protocol Test Engine**

The Safety Protocol Test Engine (SPTE) is a PC-based testing application that uses a set of input variables to configure the engine. This test engine will serve as a primary initial condition for all the DUT positive and negative test procedures for both Targets and Originators. This test engine will be the means by which communication between the Tester and DUT gets established. If the DUT is operating **without error** while interacting with this engine, the engine will run continuously.

This test engine can be configured to simulate either an originator or a target. The inputs to the test engine control features such as, originator emulation vs. target emulation, producer vs. consumer connections, multi-cast vs. single-cast, data sizes, EPI, etc. A complete list of the controls will be provided in a separate user manual.



### **F-3.2 TST1 - Standard DeviceNet Behavior**

All DeviceNet safety devices are required to execute the standard test suite of CIP tests to assure that basic CIP functionality is executed correctly. For example, DeviceNet safety devices must perform the Duplicate Mac ID test on start-up.

<b>Requirement Number</b>	<b>Requirement</b>
SRS102	Devices implementing DeviceNet Safety shall support attributes 11 of the DeviceNet object.
SRS106	The MacId attribute (if supported) in the DeviceNet object shall always reflect the value in use.
SRS110	When the switches are read, if the switches specify a valid MAC ID, (a value from 0 to 63 for MAC ID), this value shall be used as the MAC ID
SRS113	When the switches are read, if the switches used to specify the MACID are set to a valid value (a value from 0 to 63), the MACID attribute shall have Get Only access
SRS114	When the switches are read, if the switches are set to the software programmable mode (>63 for the MAC ID), the MACID attribute shall have Set access
SRS121	If the device is out-of-box it shall use the current MAC ID switch settings if these are valid values
FRS14	Each safety device shall have a single physical address that is unique on the devices network segment.
FRS15	The safety network shall not restrict any physical topologies

TST1 DeviceNet Safety Devices shall be confirmed as being compliant to Standard DeviceNet by running the standard DeviceNet conformance test suite.

#### **TST49 - DeviceNet Object Tests**

**The requirements previously allocated to TST49 are covered by the Standard conformance test for the DeviceNet object**

### **F-3.3 TST107 – Standard EtherNet/IP Behavior**

All EtherNet/IP safety devices are required to execute the standard test suite of CIP tests to assure that basic CIP functionality is executed correctly.

<b>Requirement Number</b>	<b>Requirement</b>
SRS209	Devices implementing EtherNet/IP Safety shall support attribute 7 of the TCP/IP object.
FRS14	Each safety device shall have a single physical address that is unique on the devices network segment.
FRS15	The safety network shall not restrict any physical topologies

TST107 EtherNet/IP Safety Devices shall be confirmed as being compliant to Standard EtherNet/IP by running the standard EtherNet/IP conformance test suite.

The standard EtherNet/IP test suite will confirm the support for Attribute 7 of TCP/IP object



## F-3.4 Basic Connection Establishment

This section will define tests related to establishing non-configuring connections to safety nodes. Because of the Peer-to-peer nature of Safety I/O connections, both client-side and server-side connection tests need to be defined. Client side connection tests require that the test software act as connection server, with the ability to generate various replies that will test the error handling ability of the client-side. These tests will all use the PTP test configuration.

This test plan covers both DeviceNet and EtherNet/IP DUTs. All black box tests defined in this specification (from the DUT perspective) can be run with the PTP test configuration (i.e. the operation of the bridge in these tests is not currently covered by these tests).

### F-3.4.1 Type 2 Connection Reception by Targets

#### F-3.4.1.1 TST2 - Positive Type 2 Connection Establishment Test

This test is a positive test that confirms a set of requirements by virtue of the fact that the target DUT processes and responds without error to SafetyOpen requests. No configuration functionality is confirmed in this test. The test will validate the responses received.

Requirement Number	Requirement
FRS153	All CIP Safety connections shall be established using a Forward_Open with a safety network segment (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet)
FRS163	The SCID = 0 in a Type 2b SafetyOpen shall have special meaning to indicate that the SCID shall not be checked by a target before accepting the connection.
FRS164	If the TUNID to UNID comparison is successful than the SafetyOpen destination is deemed to be correct and processing shall continue.
FRS174	Successful SafetyOpen responses shall return the connection identifiers used, as well as the Consumer_Number assigned to the originator (always 0xFFFF for single-cast connections).
FRS345	The T-to-O API, O-to-T API, and Time Correction API in a success response shall always be the same value received in the SafetyOpen
SRS2	Type 2a: When a target receives a Type 2 SafetyOpen (no configuration data) accompanied by a non-zero Safety Configuration ID (SCID = SCCRC+SCTS), it shall compare it against the SCID of the configuration and only accept the connection if they match.
SRS98	If the target has allocated identifiers from its pool, it shall insert them into the SafetyOpen success response.
SRS180	A target device that has an existing configuration shall (at a minimum) do the following when a type 2a or 2b Safety Open is received. <ul style="list-style-type: none"> <li>• If the connection path is to an output resource, verifies that the OUNID in the SafetyOpen matches the OUNID for the connection,</li> <li>• If the SCID is non-zero, confirm the SCID matches the device SCID</li> <li>• Verify that the TUNID in the SafetyOpen matches the device TUNID</li> <li>• Verify the Electronic Key matches the device</li> <li>• Verify the CPCRC over the configuration is correct,</li> <li>• Verify and validate the connection parameters, <ul style="list-style-type: none"> <li>• Validate the application path</li> <li>• Confirm the safety connection requested is supported</li> <li>• Safety parameters are within valid ranges</li> <li>• Transition the connection to the established state</li> </ul> </li> </ul>



TST2 The Type 2 Positive Connection Establishment test shall confirm that the target DUT meets a core set of requirements by virtue of the fact that successfully accepted a properly formed SafetyOpen and sent a proper formed success response.

**Required Initial Conditions:**

1. Run SPTE as an originator.

**Test Procedure:**

1. Establish a connection with an SCID equal to the DUT SCID
2. Confirm a positive SafetyOpen response is received
3. Test will inspect the SafetyOpen Response and confirm the parameters are correct
4. Confirm the O-to-T API, T-to-O API and Time Correction API all match what was sent in the SafetyOpen
5. Establish a connection with a zero SCID
6. Confirm a positive SafetyOpen response is received
7. Test will inspect the SafetyOpen Response and confirm the parameters are correct
8. Confirm the O-to-T API, T-to-O API and Time Correction API all match what was sent in the SafetyOpen
9. Test will inspect the SafetyOpen Response and confirm the parameters are correct
10. Close connection

**F-3.4.1.2 TST113 - Positive Type 2 Test for Extended Format connections**

This test is a positive test that confirms a set of requirements by virtue of the fact that the target DUT processes and responds without error to SafetyOpen requests using the Extended Format. No configuration functionality is confirmed in this test. The test will validate the responses received.

Requirement Number	Requirement
FRS153	All CIP Safety connections shall be established using a Forward_Open with a safety network segment (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet)
FRS164	If the TUNID to UNID comparison is successful than the SafetyOpen destination is deemed to be correct and processing shall continue.
FRS174	Successful SafetyOpen responses shall return the connection identifiers used, as well as the Consumer Number assigned to the originator (always 0xFFFF for single-cast connections).
FRS345	The T-to-O API, O-to-T API, and Time Correction API in a success response shall always be the same value received in the SafetyOpen
SRS2	Type 2a. When a target receives a Type 2 SafetyOpen (no configuration data) accompanied by a non-zero Safety Configuration ID (SCID = SCCRC+SCTS), it shall compare it against the SCID of the configuration and only accept the connection if they match.
SRS98	If the target has allocated identifiers from its pool, it shall insert them into the SafetyOpen success response
SRS180	A target device that has an existing configuration shall (at a minimum) do the following when a type 2a or 2b Safety Open is received. <ul style="list-style-type: none"> <li>• If the connection path is to an output resource, verifies that the OUNID in the SafetyOpen matches the OUNID for the connection,</li> </ul>



Requirement Number	Requirement
	<ul style="list-style-type: none"> <li>• If the SCID is non-zero, confirm the SCID matches the device SCID</li> <li>• Verify that the TUNID in the SafetyOpen matches the device TUNID</li> <li>• Verify the Electronic Key matches the device</li> <li>• Verify the CPCRC over the configuration is correct,</li> <li>• Verify and validate the connection parameters,                             <ul style="list-style-type: none"> <li>• Validate the application path</li> <li>• Confirm the safety connection requested is supported</li> <li>• Safety parameters are within valid ranges</li> </ul> </li> <li>• Transition the connection to the established state</li> </ul>
FRS373	All EF safety connections shall be established using the Extended format safety network segment in a Forward_Open (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet).
FRS362	When the Target of a connection using the extended format on DeviceNet, it shall use the 22-byte extended format SafetyOpen Target Application reply.
FRS363	When the Target of a connection using the extended format on EtherNet/IP, it shall use the 14-byte extended format SafetyOpen Target Application reply.

TST113 The Type 2 Extended Format Connection Establishment test shall confirm that the target DUT meets a core set of requirements by virtue of the fact that successfully accepted a properly formed SafetyOpen and sent a proper formed success response.

**Required Initial Conditions:**

1. Run SPTE as an originator of EF connections. Both DUT producing and consuming connections must be tested.
2. Test Procedure:
3. Establish a single-cast output connection (DUT is consumer) with the EF segment type and a SCID equal to the DUT SCID
4. Confirm a positive SafetyOpen response is received in the base response format
5. Test will inspect the SafetyOpen Response and confirm the parameters are correct
6. Confirm the O-to-T API, T-to-O API and Time Correction API all match what was sent in the SafetyOpen
7. Establish a Multi-cast input connection (DUT is producer) with the EF and a SCID equal to the DUT SCID
8. Confirm a positive SafetyOpen response is received with proper EF response format
9. Test will inspect the SafetyOpen Response and confirm the parameters are correct
10. Confirm the O-to-T API, T-to-O API and Time Correction API all match what was sent in the SafetyOpen
11. Test will inspect the SafetyOpen Response and confirm the parameters are correct
12. Close connection

**F-3.4.1.3 TST3 – Connection Initialization**

This test will support both the base format and Extended Format safety connection.



Requirement Number	Requirement
FRS280	After the connection is first established, the consuming application shall close base format connections if initialization is not completed within 10 seconds. The consuming application shall close Extended connections if initialization is not completed within 8.3 seconds.
FRS281	For Single-Cast, the flag Init_Complete_Out shall indicate that the first time coordination message has been received by the producer and the producer is producing data with the time stamp relative to the consumer's clock.
FRS282	For Multi-Cast, the flag Init_Complete_Out shall indicate that the first time coordination message has been received by the producer for this consumer and the producer has received the 1st time correction message based on the time coordination information.
SRS57	Safety Nodes which reside on DeviceNet shall implement the SafetyOpen and SafetyClose services of the Connection Object.

TST3 The connection establishment test shall confirm that the Consumer will close its connection if initialization isn't completed within the required time.

**Required Initial Conditions:**

1. This test will support both the base format and Extended Format safety connection.
2. Run SPTE as an originator.

**Test Procedure:**

1. If the DUT Supports Multi-cast consumer,
2. Configure the DUT so it originates a multi-cast connection request
3. Confirm the service code used 0x54
4. Reply appropriate for the connection to be established
5. Begin producing data and generate a ping count change.
6. Confirm the DUT sends a Time Coordination response
7. Continue sending data, but don't send a Time Correction message.
8. If Extended Format is used, confirm that the DUT no longer responds to Ping Count changes after 8.3 seconds OR  $(EPI * Ping\_Interval\_EPI\_Multiplier * (Timeout\_Multiplier.PI + 1))$ , whichever is less
9. If base format is used, confirm that the DUT no longer responds to Ping Count changes after 10 seconds OR  $(EPI * Ping\_Interval\_EPI\_Multiplier * (Timeout\_Multiplier.PI + 1))$ , whichever is less
10. Confirm the DUT attempts to re-establish the connection.
11. If the DUT supports Single-cast consumer
12. Configure the DUT and initiate a type 2 connection to the DUT targeting the Connection Object with service code 0x54
13. Confirm the device accepts the connection
14. Complete the SafetyOpen exchange
15. If Extended Format is used, don't produce data until after 8.3 seconds OR  $(EPI * Ping\_Interval\_EPI\_Multiplier * (Timeout\_Multiplier.PI + 1))$ , whichever is less
16. If base format is used, don't produce data until after 10 seconds OR  $(EPI * Ping\_Interval\_EPI\_Multiplier * (Timeout\_Multiplier.PI + 1))$ , whichever is less



17. Send data with ping count changes
18. Confirm the DUT does not respond with Time Coordination messages.
19. Close the connection
20. Confirm the connection must be re-established before communication continues.
21. If the DUT supports Single-cast producer
22. Configure the DUT and initiate a type 2 connection to the DUT targeting the Connection Object with service code 0x54
23. Confirm the device accepts the connection
24. Complete the SafetyOpen exchange
25. If Extended Format is used, don't respond to ping count changes until after 8.3 seconds OR  $(EPI * Ping\_Interval\_EPI\_Multiplier * (Timeout\_Multiplier.PI + 1))$ , whichever is less
26. If base format is used, don't respond to ping count changes until after 10 seconds OR  $(EPI * Ping\_Interval\_EPI\_Multiplier * (Timeout\_Multiplier.PI + 1))$ , whichever is less
27. Confirm the DUT stops producing data
28. Close the connection
29. Confirm the connection must be re-established before communication continues.

#### **F-3.4.1.4 TST4 - Connection Parameters CRC Negative Test**

This test will verify that the target detects errors via the parameters CRC in either the Base format or Extended Format SafetyOpens. Since this CRC is software generated, it is easy to simulate these errors.

Requirement Number	Requirement
SRS180	<p>A target device that has an existing configuration shall (at a minimum) do the following when a type 2a or 2b Safety Open is received</p> <p>Verify the CPCRC over the configuration is correct</p> <p>Verify that the TUNID in the SafetyOpen matches the device TUNID</p> <p>Verify the Electronic Key matches the device</p> <p>If the connection path is to an output resource, verifies that the OUNID in the SafetyOpen matches the OUNID for the connection,</p> <p>If the SCID is non-zero, confirm the SCID matches the device SCID</p> <p>verify and validate the connection parameters,</p> <p>Validate the application path</p> <p>Confirm the safety connection requested is supported</p> <p>Safety parameters are within valid ranges</p> <p>transition the connection to the established state</p>

TST4 The Connection Parameter CRC Negative test shall confirm the target DUT will detect the case when the connection parameters do not match the CPCRC value

#### **Required Initial Conditions:**

1. This test will support both Base and EF SafetyOpen Formats for this test
2. Run SPTE as an originator and Safety I/O connection supported by the DUT



**Test Procedure:**

1. Confirm the connection is established without error
2. Close the connection
3. Open the same connection, but with a CPCRC that does not match the connection parameters
4. Confirm the target DUT rejects the connection

**F-3.4.1.5 TST5 - Type 2 SCID Checking Tests**

This test will support both the base format and Extended Format safety connections.

Requirement Number	Requirement
FRS163	The SCID = 0 in a Type 2b SafetyOpen shall have special meaning to indicate that the SCID shall not be checked by a target before accepting the connection
SRS2	Type 2a: When a target receives a Type 2 SafetyOpen accompanied by a non-zero Safety Configuration ID (SCID = SCCRC+SCTS), it shall compare it against the SCID of the configuration and only accept the connection if they match.
SRS5	Also, Safety Devices shall NV-store the signature for the configuration (SCID) in the device.

TST5 The Type 2 with matching SCID Test shall confirm that the DUT correctly reacts to these under the various SCID conditions.

**Required Initial Conditions:**

1. This test will support both the base format and Extended Format safety connections
2. Run SPTE as an originator.

**Test Procedure:**

1. Configure the DUT with a representative configuration and SCID to establish initial conditions and ownership
2. Close all connections
3. Issue a SafetyOpen **without** configuration data and a different Time Stamp in the SCID with owner OUNID
4. Confirm the device rejects the connection
5. Issue a SafetyOpen **without** configuration data and a different Time Stamp in the SCID with a different OUNID
6. Confirm the device rejects the connection
7. Issue a SafetyOpen without configuration, but with an incorrect CRC in the SCID with owners OUND
8. Confirm the device rejects the connection
9. Issue a SafetyOpen without configuration, but with an incorrect CRC in the SCID with different OUND
10. Confirm the device rejects the connection



If the connection resource is NOT an output, execute the following 2 steps.

1. Issue a SafetyOpen with no configuration and an SCID =0 with different OUNID
2. Confirm the device accepts the connection.

#### F-3.4.1.6 TST6 - Electronic Key Mismatch test

These tests will generate requests with incorrect keying information and verify the target rejects the connection. This test will support both the base format and the Extended Format safety connections.

Requirement Number	Requirement
FRS342	The Electronic Key shall always be processed by Target devices with integrity (cannot be confirmed externally)
FRS344	The compatibility bit and behavior in the Electronic Key shall be supported as defined by safety originators and targets, wildcards (fields set to 0) for individual fields within the Electronic Key shall not be supported

TST6 The Electronic Key Mismatch test shall confirm that when a SafetyOpen is sent with a non-matching key is rejected by the target.

##### Required Initial Conditions:

1. This test will support both the base format and the Extended Format safety connections.
2. Run Test Engine.

##### Test Procedure:

1. Issue a series of SafetyOpens, each with one of the electronic key parameters set to an invalid value and the compatibility bit in the Major Rev set to 0
2. Invalid vendor Id
3. Invalid Product Type
4. Invalid Product Code
5. Invalid Major Rev
6. Invalid Minor Rev
7. Confirm the SafetyOpen is rejected
8. Issue a series of SafetyOpens, each with one of the electronic key parameters set to a zero value and the compatibility bit in the Major Rev set to 0
9. Invalid vendor Id
10. Invalid Product Type
11. Invalid Product Code
12. Invalid Major Rev
13. Invalid Minor Rev
14. Confirm each SafetyOpen is rejected
15. Issue a SafetyOpen, with a **matching** electronic key and compatibility bit in the major revision set to 0 (all must match)
16. Confirm the SafetyOpen is accepted



### F-3.4.1.7 TST7 –Target Connection Id Allocation Tests

This test will support both the base format and the Extended Format safety connections

Requirement Number	Requirement
SRS95	During connection processing of a SafetyOpen DeviceNet safety identifiers shall be assigned from the available pool of DeviceNet Group 1 identifiers
SRS96	Safety connection originators shall initially ask the target (via SafetyOpen) to allocate one, two, or three identifiers according to the rules stated in Table 24 (This is accomplished by inserting 0xFFFF identifiers in the SafetyOpen request sent to the target)
SRS97	If a safety target cannot or refuses to allocate the identifier(s), the target shall respond to the Safety Open request with a 0x01 general error status with extended status 0x031F indicating no connection resources are available.
SRS99	In order to provide consistency among producers and consumers, the following rule shall be used when allocating identifiers: The first MSGID to be allocated shall start at the highest available value and move downward.
SRS100	Multi-Cast producers must allocate an identifier from its own pool, i.e. the source MACID must be that of the producer. Multi-cast producers shall reject any connection request that attempts to assign IDs for its produced data or time correction data
SRS179	When a safety device closes a connection in which group 1 IDs were allocated to another device, the group 1 ID shall be placed into quarantine for a period equal to the $(\text{connection\_EPI} * \text{Ping\_Interval\_Multiplier}) * (\text{Timeout\_Multiplier.PI} + 2)$ . During this time, the ID cannot be used or allocated to another device.

TST7 The Target Connection Id Allocation test shall confirm proper allocation rules are being followed.

#### Device Capability Requirements:

To execute this test, it assumes that it is possible to create a sufficient number of connections to deplete the available number of Group 1 identifiers. This test therefore requires the device supports at least 1 single-cast and 1 multi-cast with at least 8 consumers. If a device does not support these capabilities, the quarantine requirement (SRS179) won't be confirmed. Devices which can't be tested must take responsibility to independently confirm they are quarantining identifiers properly. This test will support both the base format and the Extended Format safety connections

#### Required Initial Conditions:

1. Run Test Engine.

#### Test Procedure:

1. Commission and Configure the DUT with a representative configuration and SCID to establish initial conditions
2. Issue a SafetyOpen to the DUT for one of each type of safety service the DUT supports. All supported services should be tested
3. When connecting to a Single-cast output, fill in **all identifiers** with 0xFFFF to request the Target allocate all the Ids
4. When connecting to a Multi-cast input, provide an identifier for the tester production and fill in 0xFFFF for all identifiers the Target will produce on



5. Confirm the Ids are allocated from the highest range of Group 1 Ids
6. If the device supports more connections, continue to open additional connections until the Target has allocated all available Ids.
7. Confirm that after the Ids are used up, the DUT responds with an error 0x01 extended code 0x031F
8. Generate a CRC error on one of the single-cast output connections so the target closes the connection and puts the group 1 Ids in quarantine. Note the connection Ids that were used
9. During the quarantine period, open another connection (to a different resource if possible) and request all the Ids.
10. Confirm the DUT does not allocate the IDs that should be in quarantine

#### **F-3.4.1.8 TST8 - Multi-cast Producer, Consumer Number Allocation**

This test will make sure Multi-cast producers quarantine consumer numbers for the required period of time. This test will support both the base format and the Extended Format safety connections

Requirement Number	Requirement
FRS325	The Validator Connection Establishment Engine shall ensure that the Consumer_Open resource for any allocated consumer number is set to Closed for a minimum time of (Timeout_Multiplier.PI+2) * Ping Interval prior to re-allocating that Consumer number to another connection. This is to ensure that any previously established connections have timed out prior to allocating the number to a new connection.

TST8 The Multi-cast Producer Consumer\_Number Allocation test shall confirm that the producer quarantines consumer numbers properly.

##### **Required Initial Conditions:**

1. This test will support both the base format and the Extended Format safety connections
2. Run Test Engine.
3. Input: Max\_number\_of\_consumers supported by the device

##### **Test Procedure:**

1. The test shall establish consumer connection with producer DUT so max consumers supported are in use.
2. The tester will log the consumer numbers allocated to each consumer.
3. The tester will stop responding to ping requests on 1 consumer (note the consumer #)
4. The tester will wait until the producer detects the failure and closes the connection.
5. The tester will attempt to re-establish the connection using another Node Id, but before the Timeout\_Multiplier.PI+2 ping intervals has gone by.
6. The tester shall confirm that the producer DUT rejects the connection
7. The tester will repeat the connection attempt after the Timeout\_Multiplier.PI + 2 Ping Intervals has gone by



#### F-3.4.1.9 TST99 – DeviceNet Base Format Multi-cast Producer Time Correction Connections

When the DeviceNet protocol is used for the multi-cast connection type the time correction section is not part of the data message. The data section and time stamp section are transmitted on a separate link and the time correction data is transmitted on a separate link. This test confirms that the base format **Target** Multi-cast producer on DeviceNet responds to a SafetyOpen with the proper number of connections.

Requirement Number	Requirement
FRS29	For devices on a DeviceNet network, the Multi-cast Time_Correction_Value shall be sent to the consumers via a separate multi-cast link connection.
FRS92	For DeviceNet Multi-cast connections, a separate message shall be used to communicate time correction information from the produce to each multi-cast consumer.

TST99 The Multi-cast Producer Time Correction Connection test shall confirm that the base format multi-cast producer DUT will generate proper connection resources.

##### Required Initial Conditions:

1. Run the SPTE as Client

##### Test Procedure:

1. The SPTE sends a safety open to the DeviceNet Producer DUT
2. The DUT will receive the safety open.
3. The test confirms the DUT sends a valid safety open response of the proper type.
4. The test shall confirm the DUT sets up 3 separate connections.
5. The DUT will send the 1<sup>st</sup> data message over the 1st link. The data message is the data section and time stamp section.
6. The test will receive the 1<sup>st</sup> data message and verify the data message composition and Ping Request
7. The test will send a valid time coordination message over the 2<sup>nd</sup> link.
8. The DUT will receive the time coordination message and send a time correction message over the 3rd link.
9. The test will receive the time correction message and verify the data.

#### F-3.4.1.10 TST120 – DeviceNet Extended Format Multi-cast Producer Time Correction Connections and Rollover Seeding

When the DeviceNet protocol is used for the Extended Format multi-cast connection type the time correction section is a special format not part of the data message. The data and time stamp are transmitted on a separate link and the time correction data is transmitted on a separate link. This test confirms that the Extended Format **Target** Multi-cast producer on DeviceNet responds to a SafetyOpen with the proper number of connections and confirms the proper format for the Time Correction message.



<b>Requirement Number</b>	<b>Requirement</b>
FRS29	For devices on a DeviceNet network, the Multi-cast Time Correction Value shall be sent to the consumers via a separate mutli-cast link connection.
FRS92	For DeviceNet Multi-cast connections, a separate message shall be used to communicate time correction information form the produce to each multi-cast consumer.
FRS371	When a connection is established with the Extended format, the EF Time Correction format shall be used; otherwise the base format shall be used.
FRS378	The active seeding of the CRC with the rollover count in Extended Format Multi-cast messages shall begin immediately on first production.

TST120 The Extended Format Multi-cast Producer Time Correction Connection test shall confirm that the Extended Format multi-cast producer DUT will generate proper connection resources with the proper formats on DeviceNet.

**Required Initial Conditions:**

1. Run the SPTE as Client

**Test Procedure:**

1. The SPTE sends a SafetyOpen for a Multi-cast input (DUT produces data) with the proper segment type to the DeviceNet Producer DUT
2. The DUT will receive the safety open.
3. The test confirms the DUT sends a valid safety open response of the proper type.
4. The test shall confirm the DUT sets up 3 separate connections.
5. The DUT will send the 1<sup>st</sup> data message over the 1st link. The data message is the data and time stamp section with the initial rollover count included in the CRC seeding.
6. The test will receive the 1<sup>st</sup> data message and verify the data message format, CRC, and Ping Request
7. The test will send a valid Extended Format time coordination message over the 2<sup>nd</sup> link.
8. The DUT will receive the Extended Format time coordination message and send a Extended Format time correction message over the 3rd link.
9. The test will receive the Extended Format time correction message and verify the data.

**F-3.4.2 Type 2 Connection Generation by Originators**

**F-3.4.2.1 TST9 - Positive Type 2 Single-Cast Connections on DeviceNet**

This test is a positive test that confirms that the originator DUT on DeviceNet generates a proper SafetyOpen request without configuration data. No configuration functionality is confirmed in this test. The test will validate the SafetyOpen packet generated. This test will support both the base format and the Extended Format safety connection.

<b>Requirement Number</b>	<b>Requirement</b>
FRS153	All CIP Safety connections shall be established using a Forward Open with a safety network segment (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet)
FRS166	To establish ownership of the configuration (and the connection in outputs), in targets the OUNID of the originating device shal. be passed to the target in the SafetyOpen service.



<b>Requirement Number</b>	<b>Requirement</b>
FRS170	DeviceNet originators shall use the Connection Object's Safety Open request for local targets
SRS57	Safety Nodes which reside on DeviceNet shall implement the SafetyOpen and SafetyClose services of the Connection Object.
SRS96	Safety connection originators shall initially ask the target (via SafetyOpen) to allocate one, two, or three identifiers. (This is accomplished by inserting 0xFFFFFFFF identifiers in the SafetyOpen request sent to the target)
FRS373	All Extended Format safety connections shall be established using the Extended format safety network segment in a Forward_Open (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet).
FRS379	When Extended Format single-cast connections are originated by the producer (i.e outputs), the target consumer shall provide initialization values in the SafetyOpen Response

**TST9** The Positive Type 2 Connection Generation Test shall confirm that the originator DUT generates a proper DeviceNet Single-Cast SafetyOpen message for both the EF and base format .

**Required Initial Conditions:**

1. Configure the DeviceNet originator DUT with a single-cast I/O connection and representative configuration. This test will support both the base format and the Extended Format safety connection
2. Run SPTE as server. Input the connection parameters configured in the originator

**Test Procedure:**

1. Confirm the DUT opens a Class 3 connection
2. The tester accepts this connection
3. Confirm the DUT issues the SafetyOpen by requesting the target provide all the Ids
4. Test will confirm a SafetyOpen is sent over the class 3 connection with service code 0x54
5. Test will confirm the service message is routed to the connection object
6. Test will inspect the SafetyOpen and confirm the parameters are correct
7. Test will confirm the CPCRC is correct
8. Test will confirm the safety segment type is correct for the format configured in the originator
9. Test will confirm the OUNID matches the originator
10. Test will confirm the TUNID matches the configured value
11. Test will confirm the Electronic Key matches the configured value
12. Test will confirm the Connection Parameters match the configured values
13. Test will confirm the Time Correction parameters are null values
14. If the segment type is the Extended Format, the test will confirm the Initial Time Stamp and Initial Rollover value are set to 0xFFFF
15. Test will reply with a success SafetyOpen Response of the proper type and values



16. Test will confirm the DUT accepts the response and the safety connection is established

#### **F-3.4.2.2 TST118 - Positive Type 2 Single-Cast Connections on EtherNet/IP**

This test is a positive test that confirms that the originator DUT on EtherNet/IP generates a proper Safety Forward Open request without configuration data. No configuration functionality is confirmed in this test. The test will validate the Forward Open packet generated. This test will support both the base format and the Extended Format safety connection.

<b>Requirement Number</b>	<b>Requirement</b>
FRS153	All CIP Safety connections shall be established using a Forward_Open with a safety network segment (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet)
FRS166	To establish ownership of the configuration (and the connection in outputs), in targets the OUNID of the originating device shall be passed to the target in the SafetyOpen service
FRS373	All Extended Format safety connections shall be established using the Extended format safety network segment in a Forward_Open (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet).
FRS379	When Extended Format single-cast connections are originated by the producer (i.e outputs), the target consumer shall provide initialization values in the SafetyOpen Response.

TST118 The Positive Type 2 Connection Generation Test shall confirm that the originator DUT generates a proper EtherNet/IP Single-Cast SafetyOpen message for both the EF and base format

#### **Required Initial Conditions:**

1. Configure the EtherNet/IP originator DUT with a single-cast I/O connection and representative configuration. This test will support both the base format and the Extended Format safety connection
2. Run SPTE as server. Input the connection parameters configured in the originator

#### **Test Procedure:**

1. Confirm the DUT issues the Forward Open with the proper safety segment
2. Test will confirm a Forward Open with service code 0x54
3. Test will confirm the service message is routed to the connection manager object
4. Test will inspect the Forward Open and confirm the parameters are correct
5. Test will confirm the CPCRC is correct
6. Test will confirm the safety segment type is correct for the format configured in the originator
7. Test will confirm the OUNID matches the originator
8. Test will confirm the TUNID matches the configured value
9. Test will confirm the Electronic Key matches the configured value
10. Test will confirm the Connection Parameters match the configured values
11. Test will confirm the Time Correction parameters are default values



12. If the segment type is the Extended Format, the test will confirm the Initial Time Stamp and Initial Rollover value are set to 0xFFFF
13. Test will reply with a success SafetyOpen Response of the proper type and values
14. Test will confirm the DUT accepts the response and the safety connection is established

#### F-3.4.2.3 TST10 - Positive Type 2 Multi-cast Connections on DeviceNet

This test is a positive test that confirms that the **originator** DUT on DeviceNet generates a proper Multi-cast SafetyOpen request without configuration data. No configuration functionality is confirmed in this test. The test will validate the SafetyOpen packet generated. This test will support both the base format and the Extended Format safety connection.

Requirement Number	Requirement
FRS29	For devices on a DeviceNet network, the Multi cast Time_Correction_Value shall be sent to the consumers via a separate multi-cast transport connection.
FRS153	All CIP Safety connections shall be established using a Forward_Open with a safety network segment (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet)
FRS170	DeviceNet originators shall use the Connection Object's Safety Open request for local targets
FRS373	All Extended Format safety connections shall be established using the Extended format safety network segment in a Forward_Open (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet).
FRS377	When Extended Format multi-cast connections are originated by the consumer (i.e. multi-cast inputs); the target producer shall provide initialization values in the SafetyOpen Response.
SRS57	Safety Nodes which reside on DeviceNet shall implement the SafetyOpen and SafetyClose services of the Connection Object.
SRS95	During connection processing of a SafetyOpen DeviceNet safety identifiers shall be assigned from the available pool of DeviceNet Group 1 identifiers
SRS96	Safety connection originators shall initially ask the target (via SafetyOpen) to allocate one, two, or three identifiers. (This is accomplished by inserting 0xFFFFFFFF identifiers in the SafetyOpen request sent to the target)
SRS99	In order to provide consistency among producers and consumers, the following rule shall be used when allocating identifiers: The first MSGID to be allocated shall start at the highest available value and move downward.

TST10 The Positive Type 2 Multi-cast Connection Generation on DeviceNet test shall confirm the originator DUT will generate a proper connection request for both base and Extended Formats.

#### Required Initial Conditions:

1. Configure the DeviceNet originator DUT with a Multi-cast consumer I/O connection and representative configuration for either the base format or Extended Format
2. Run SPTE as server. Input the connection parameters configured in the originator

#### Test Procedure:

1. Confirm the DUT opens a Class 3 connection
2. The tester accepts this connection



3. Confirm the DUT issues the SafetyOpen with proper safety segment and by providing Ids for its producing connections, but requests Ids for the targets producing connections
4. Confirm that the allocated Ids are from the high-end of the group 1 range
5. Test will confirm a SafetyOpen is sent over the class 3 connection with service code 0x54
6. Test will confirm the service message is routed to the connection object
7. Test will inspect the SafetyOpen and confirm the parameters are correct
8. Test will confirm the CPCRC is correct
9. Test will confirm the safety segment type is correct for the format configured in the originator
10. Test will confirm the OUNID matches the originator
11. Test will confirm the TUNID matches the configured value
12. Test will confirm the Electronic Key matches the configured value
13. Test will confirm the Connection Parameters match the configured values
14. Test will confirm that the Time Correction connection is included
15. If the segment type is the Extended Format, the test will confirm the Initial Time Stamp and Initial Rollover value are set to 0xFFFF
16. Test will reply with a success SafetyOpen Response of the proper type and values
17. Test will confirm the DUT accepts the response and the safety connection is established

#### **F-3.4.2.4 TST119 - Positive Type 2 Multi-cast Connections on EtherNet/IP**

This test is a positive test that confirms that the originator DUT on EtherNet /IP generates a proper Multi-cast SafetyOpen request without configuration data. No configuration functionality is confirmed in this test. The test will validate the Safety Forward Open packet generated. This test will support both the base format and the Extended Format safety connection

<b>Requirement Number</b>	<b>Requirement</b>
FRS30	For devices on a non-DeviceNet network, the data and Time_Correction_Value shall be concatenated and sent to the consumers via a single multi-cast transport connection.
FRS153	All CIP Safety connections shall be established using a Forward_Open with a safety network segment (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet)
FRS373	All Extended Format safety connections shall be established using the Extended format safety network segment in a Forward_Open (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet).
FRS377	When Extended Format multi-cast connections are originated by the consumer (i.e. multi-cast inputs); the target producer shall provide initialization values in the SafetyOpen Response.

TST119 The Positive Type 2 Multi-cast Connection Generation on EtherNet/IP test shall confirm the originator DUT will generate a proper connection request for both base and Extended Formats.

#### **Required Initial Conditions:**

1. Configure the EtherNet/IP originator DUT with a Multi-cast consumer I/O connection and representative configuration for either the base format or Extended Format



2. Run SPTE as server. Input the connection parameters configured in the originator

**Test Procedure:**

1. Confirm the DUT issues the SafetyOpen with the proper safety segment
2. Test will confirm a SafetyOpen is sent with service code 0x54
3. Test will confirm the service message is routed to the connection manager object
4. Test will inspect the SafetyOpen and confirm the parameters are correct
5. Test will confirm the CPCRC is correct
6. Test will confirm the safety segment type is correct for the format configured in the originator
7. Test will confirm the OUNID matches the originator
8. Test will confirm the TUNID matches the configured value
9. Test will confirm the Electronic Key matches the configured value
10. Test will confirm the Connection Parameters match the configured values
11. Test will confirm that the Time Correction parameters are the default values
12. If the segment type is the Extended Format, the test will confirm the Initial Time Stamp and Initial Rollover value are set to 0xFFFF
13. Test will reply with a success SafetyOpen Response of the proper type and values
14. Test will confirm the DUT accepts the response and the safety connection is established

**F-3.4.2.5 TST114- Positive Extended Format Type 2 Single-Cast Connection Origination**

This test is a positive test that confirms that the originator DUT generates a proper SafetyOpen request with the correct segment type without configuration data. No configuration functionality is confirmed in this test. The test will validate the SafetyOpen packet generated. This procedure is written so that it can be applied to either EtherNet/IP or DeviceNet.

Requirement Number	Requirement
FRS153	All CIP Safety connections shall be established using a Forward_Open with a safety network segment (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet)
FRS166	To establish ownership of the configuration (and the connection in outputs), in targets the OUNID of the originating device shall be passed to the target in the SafetyOpen service.
FRS373	All Extended Format safety connections shall be established using the Extended format safety network segment in a Forward_Open (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet).

TST114 The Positive Extended Format Type 2 Connection Generation Test shall confirm that the originator DUT generates a proper Extended Format Single-Cast SafetyOpen message

**Required Initial Conditions:**

1. Configure the originator DUT with a Extended Format single-cast I/O connection and representative configuration
2. Run SPTE as server. Input the connection parameters configured in the originator



**Test Procedure:**

1. Confirm the DUT sends the SafetyOpen with the Extended Format
2. The tester accepts this connection
3. Test will confirm the service message is routed to the connection manager object if the DUT is on EtherNet/IP or over an explicit connection to the connection object if the DUT is on DeviceNet
4. Test will inspect the SafetyOpen and confirm the parameters are correct
5. Test will confirm the CPCRC is correct
6. Test will confirm the safety segment is present
7. Test will confirm the OUNID matches the originator
8. Test will confirm the TUNID matches the configured value
9. Test will confirm the Electronic Key matches the configured value
10. Test will confirm the Connection Parameters match the configured values
11. Test will confirm the Time Correction parameters are null values
12. Test will confirm the Initial Time Stamp and Initial Rollover Value set to values other than 0xFFFF
13. Test will reply with a success standard SafetyOpen Response
14. Test will confirm the DUT accepts the response and the safety connection is established

**F-3.4.2.6 TST115- Positive Extended Format Type 2 Multi-Cast Connection Origination**

This test is a positive test that confirms that the originator DUT generates a proper Multi-cast SafetyOpen request with the correct segment type without configuration data. No configuration functionality is confirmed in this test. The test will validate the SafetyOpen packet generated. This procedure is written so that it can be applied to either EtherNet/IP or DeviceNet.

Requirement Number	Requirement
FRS153	All CIP Safety connections shall be established using a Forward_Open with a safety network segment (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet)
FRS166	To establish ownership of the configuration (and the connection in outputs), in targets the OUNID of the originating device shall be passed to the target in the SafetyOpen service.
FRS373	All Extended Format safety connections shall be established using the Extended format safety network segment in a Forward_Open (EtherNet/IP, SERCOS III) or a SafetyOpen (DeviceNet).

TST115 The Positive Extended Format Type 2 Multi-cast Connection Generation Test shall confirm that the originator DUT generates a proper Extended Format Multi-Cast SafetyOpen message and accepts the Extended Format response

**Required Initial Conditions:**

1. Configure the originator DUT with a Extended Format Multi-cast I/O connection and representative configuration
2. Run SPTE as server. Input the connection parameters configured in the originator



**Test Procedure:**

1. Confirm the DUT sends the SafetyOpen with the Extended Format
2. The tester accepts this connection
3. Test will confirm the service message is routed to the connection manager object if the DUT is on EtherNet/IP or over an explicit connection to the connection object if the DUT is on DeviceNet
4. Test will inspect the SafetyOpen and confirm the parameters are correct
5. Test will confirm the CPCRC is correct
6. Test will confirm the safety segment is present
7. Test will confirm the OUNID matches the originator
8. Test will confirm the TUNID matches the configured value
9. Test will confirm the Electronic Key matches the configured value
10. Test will confirm the Connection Parameters match the configured values
11. Test will confirm the Time Correction parameters are null values
12. Test will confirm the Initial Time Stamp and Initial Rollover Value set to values of 0xFFFF
13. Test will reply with a success Extended Format SafetyOpen Response
14. Test will confirm the DUT accepts the response and the safety connection is established

**F-3.4.2.7 TST100 Electronic Key Generation Test**

Requirement Number	Requirement
FRS341	The Electronic Key segment shall be present in all SafetyOpen and Safety Forward Open service requests.

TST100 The Electronic Key Generation Tests shall confirm that originators provide a valid Electronic Key segment with every SafetyOpen or Safety Forward Open.

**Required Initial Conditions:**

1. This test will support both the base and Extended Format type SafetyOpen or Safety Forward Open and will operate over DeviceNet or EtherNet/IP
2. Commission and Configure the DUT with a representative configuration to establish initial conditions. Configure Single cast targets and multi-cast producing targets.
3. Run SPTE as server. Input the connection parameters configured in the originator

**Test Procedure:**

1. Confirm the originator provides any electronic key segment as part of the SafetyOpen or Safety Forward Open

**F-3.4.3 SafetyClose Tests**

**F-3.4.3.1 TST101 SafetyClose Processing by Targets**

Requirement Number	Requirement
FRS337	Targets receiving the SafetyClose service request shall close the connection who's Originator Vendor ID, Connection Serial Number, and Originator Serial Number



Requirement Number	Requirement
	parameters match. Additional information provided by this service (ie, Connection Path) shall be ignored by the target.
FRS335	The SafetyClose request shall cause all resources in all nodes participating in the connection to be deallocated, including connection IDs, bandwidth, and internal memory buffers.

TST101 The SafetyClose processing by Targets test shall verify that targets accept the message and properly close the associated connection.

**Required Initial Conditions:**

1. Commission and Configure the DUT with a representative configuration to establish initial conditions.
2. This test will support either base format or Extended Format connections

**Test Procedure:**

1. Establish connection of the DUT.
2. Complete all Time Coordination steps to completely establish the connection
3. Send a Safety Close with the Originator Vendor Id, Connection Serial Number, and Originator Serial Number used in the initial SafetyOpen
4. Confirm the device sends a success response
5. Establish connection of the DUT.
6. Complete all Time Coordination steps to completely establish the connection
7. Send a Safety Close with different Connection Serial Number used in the initial SafetyOpen
8. Confirm the device sends an error response

**F-3.5 Common Run-Time Tests**

This section shall define a core series of tests that will be referenced in the test procedures of other black box tests. These tests form a core library of scripts that are common to all run-time messaging.

After a connection has been established, the positive test engines will maintain connections if the data messages are generated correctly and transmitted/received within the timing parameters. Safety devices can be producers, consumers, or both. The safety device connections can be single-cast or multi-cast.

The common positive-behavior tests are organized by functionality where applicable.

1. A producer generates and transmits data messages and receives time coordination messages.
  - a. multi-cast connection
  - b. single-cast connection
2. A consumer receives data messages and generates and transmits time coordination messages.
  - a. multi-cast connection



- b. single-cast connection

Some requirements are independent of the connection type or device type. Data message size, safety CRC usage, etc. These requirements are organized as separate tests.

### **F-3.5.1 Positive Producer Tests**

When testing a DUT producer the tester will be the consumer. The tester will generate time coordination messages, verify any data message received. These tests will be used by both the originator and target positive test engines.

#### **F-3.5.1.1 TST13 – Producer CRC & PID/CID Test**

The Producer ID (PID) is the initial seed value for the safety CRC's of all messages from the data producer of a connection. The Consumer ID (CID) is the initial seed value for the safety CRC's of all messages from the consumer of a connection. Every message that is transmitted or received will have the safety CRC verified.

<b>Requirement Number</b>	<b>Requirement</b>
FRS32	All safety messages and safety response messages shall be encoded with the safety CRC's defined in Appendix E
FRS155	The Producer Identifier (PID) shall be used as an initial seed value in the CRCs in the runtime protocol.
FRS330	The Consumer Identifier (CID) shall be used as an initial seed value in the CRC in the Time Coordination message.
FRS331	The Producer shall obtain the Consumer Identifier from either the SafetyOpen or the SafetyOpen Response (depending upon originator) and use the value as an initial seed in runtime checking the safety CRCs
SRS152	During connection establishment the producing and consuming nodes shall exchange their respective PID/CID information (Vendor ID + Device Serial Number + Connection Serial Number). The originator sends its ID in the SafetyOpen request, and the target returns its ID as part of the SafetyOpen Success response.
SRS178	The PID shall be used to seed Data production CRCs and Time Correction CRC, and the CID shall be used to seed the Time Coordination CRC

TST13 The Producer CRC & PID/CID Run-Time Test shall be a positive test to confirm that the Producer DUT is both creating proper safety CRCs with the PID and interpreting valid safety CRCs with the CID.

#### **Required Initial Conditions:**

1. This test will apply to both base and Extended Format connections
2. A connection must exist between the test and the Producer DUT. The connection process has previously exchanged the PID and CID between the originator and the target.

#### **Test Procedure:**

1. All messages received from the DUT will have the safety CRC's recalculated with the PID and the received data.
2. The safety CRC's will be verified to be equal.
3. All messages transmitted to the DUT will have the safety CRC's calculated with the CID by the test.



This test will not generate CRC error conditions or confirm such detection at the DUT. While this test is running, no error or faults should be generated by the DUT.

### **F-3.5.1.2 Producer Data Message Generation**

The tests in this section will verify the requirements that the data section of the data message is generated correctly. The data section is common to devices and connection types. These tests will use other tests that check the CRC's are calculated using the PID and mode byte. Test 11 through 13 will be executed on all transmitted DUT data messages.

#### **F-3.5.1.2.1 TST14 – Base Format Producer - 1 to 2 Bytes Data**

When the base format data size is 1 or 2 bytes the data section consists of data, mode byte, data CRC and complemented data CRC.

<b>Requirement Number</b>	<b>Requirement</b>
FRS38	The 1 or 2 Byte Data section for the base format shall consist of the actual data byte or bytes, the mode byte, the actual CRC, and the complement CRC.
FRS39	The Base format 1 or 2 Byte Actual Data CRC calculation shall include: Producer Identifier (PID) (by CRC-S1) The (Mode Byte) AND (0xE0) (by CRC-S1) Actual Data byte(s) (by CRC-S1)
FRS40	The Base Format 1 or 2 Byte Complement Data CRC calculation shall include: Producer Identifier (PID) (by CRC-S1) The (Mode Byte XOR 0xFF) AND (0xE0) (by CRC-S2) Actual Data bytes XOR 0xFF (by CRC-S2)

TST14 The Base format Producer Packet Generation - 1 to 2 Bytes Data - Positive test shall confirm that the producer is generating the 1-2 byte data packet correctly.

#### **Required Initial Conditions:**

1. A safety I/O connection must exist between the test and the Producer DUT. The connection process has previously exchanged the PID and CID between the originator and the target.
2. A data message is received by the tester.

#### **Test Procedure:**

1. The data section of the data message is verified to be built correctly.
2. Run TST13, Producer CRC & PID/CID Run-Time Test
3. The test complements the received data and recalculates the complemented data CRC. The calculated complemented data CRC should be equal to the received complemented data CRC.



### **F-3.5.1.2.2 TST109 – Extended Producer – 1 or 2 Bytes Data**

When the Extended format data size is 1 or 2 bytes the data section consists of data, mode byte, Time Stamp and data CRC.

<b>Requirement Number</b>	<b>Requirement</b>
FRS365	The 1 or 2 Byte Data section for the Extended format shall consist of the actual data byte or bytes, the mode byte, the time stamp, and CRC-S5 calculated over the entire packet.
FRS366	The Extended 1 or 2 Byte Data CRC calculation shall use CRC-S5 and include: Producer Identifier (PID) 16-bit Rollover count Mode Byte & 0xE0 Actual Data byte(s) Time Stamp

**TST109** The Extended format Producer Packet Generation - 1 to 2 Bytes Data - Positive test shall confirm that the producer is generating the 1-2 byte data packet correctly.

#### **Required Initial Conditions:**

1. An EF safety I/O connection must exist between the test and the Producer DUT. The connection process has previously exchanged the PID, CID and roll-over counts between the originator and the target.
2. A data message is received by the tester.

#### **Test Procedure:**

1. The data section of the data message is verified to be built correctly.
2. Run TST13, Producer CRC & PID/CID Run-Time Test
3. The test calculates the data CRC to include all the components identified in FRS366.

### **F-3.5.1.2.3 TST15 – Base Format Producer - 3 to 250 Bytes Data**

When the data size is 3 to 250 bytes the data section consist of data, mode byte, data CRC, complemented data and complemented data CRC.

<b>Requirement Number</b>	<b>Requirement</b>
FRS41	The Base format 3 to 250 Byte Data section shall consist of the actual data bytes, the mode byte, the actual CRC, the complement data bytes, and the complement CRC.
FRS42	The Base format 3-to250 Byte Actual Data CRC calculation shall include: Producer Identifier (PID) (by CRC-S3) The (Mode Byte) AND (0xE0) (by CRC-S3) Actual Data byte(s) (by CRC-S3)
FRS43	The Base format 3 to 250 Byte Complement Data CRC calculation shall include: Producer Identifier (PID) (by CRC-S3) The (Mode Byte XOR 0xFF) AND (0xE0) (by CRC-S3) <b>Complemented</b> Data bytes (by CRC-S3)

**TST15** The Base Format Producer Packet Generation – 3 to 250 Byte Data – Positive test shall confirm that the producer is generating the 3 to 250 byte data packets correctly.



**Required Initial Conditions:**

1. A safety I/O connection must exist between the test and the Producer DUT. The connection process has previously exchanged the PID and CID between the originator and the target.
2. A data message is received by the tester.

**Test Procedure:**

1. The data section of the data message is verified to be built correctly.
2. Run TST13, Producer CRC & PID/CID Run-Time Test
3. The test complements the received data and compares the received complemented data to the test complemented data. The data comparisons should pass.
4. The test calculates the complemented data CRC using the received complemented data. The calculated complemented data CRC should be equal to the received complemented data CRC.

**F-3.5.1.2.4 TST110 – Extended Format Producer - 3 to 250 Bytes Data**

This test is a positive test to confirm proper operation of a Extended Format producer DUT for data sizes greater than 2 bytes.

Requirement Number	Requirement
FRS367	The Extended 3 to 250 Byte Data section shall consist of the actual data bytes, the mode byte, CRC-S3, the complement data bytes, the time stamp and CRC-S5
FRS368	The Extended format 3-to250 Byte Actual Data CRC calculation shall include: Producer Identifier (PID) (by CRC-S3) Rollover Count (by CRC-S3) The (Mode Byte) AND (0xE0) (by CRC-S3) Actual Data byte(s) (by CRC-S3)
FRS369	The Extended format 3-to250 Byte Complement Data CRC calculation shall include: Producer Identifier (PID) (by CRC-S5) Rollover Count (by CRC-S5) Mode Byte AND 0x1F (by CRC-S5) <b>Complemented</b> Data bytes (by CRC-S5) Time Stamp (by CRC-S5)

TST110 The Extended Format Producer Packet Generation – 3 to 250 Byte Data – Positive test shall confirm that the producer is generating the 3 to 250 byte data packets correctly

**Required Initial Conditions:**

1. An Extended Format safety I/O connection must exist between the test and the Producer DUT. The connection process has previously exchanged the PID, CID, and Roll-over count between the originator and the target.
2. A data message is received by the tester.

**Test Procedure:**

1. The data section of the data message is verified to be built correctly.
2. Run TST13, Producer CRC & PID/CID Run-Time Test
3. The test complements the received data and compares the received complemented data to the test complemented data. The data comparisons should pass.



4. The test calculates the actual data CRC using the received data, PID and rollover count. The CRC should match the received value
5. The test calculates the complemented data CRC using the PID, rollover count, complemented data and Time Stamp,. The complement CRC should match the values sent.

#### **F-3.5.1.3 TST16 - Producer Packet Generation Mode Byte**

<b>Requirement Number</b>	<b>Requirement</b>
FRS126	The TBD_Bit Shall be set to 0 by the Safety Validator
FRS183	The N_Run_Idle complement bit shall always be the complement of the Run_Idle bit
FRS191	The N TBD_Bit complement bit shall always be the complement of the TBD_Bit

TST16 The Producer Packet Generation Mode Byte – Positive Test shall confirm that the producer is generating the Mode byte correctly.

##### **Required Initial Conditions:**

1. This test will function for both base format and Extended Format connections
2. A safety I/O connection must exist between the test and the Producer DUT.
3. A data message is received by the tester.

##### **Test Procedure:**

1. The TBD\_bit is verified to be set to 0.
2. The N\_TBD\_bit is verified to be set to 1.
3. The complement of Run\_Idle bit is compared to the received N\_Run\_Idle bit. The values should be equal.

#### **F-3.5.1.4 TST17 – Base Format Producer Packet Time Stamp CRC**

This test will verify the requirements that the time stamp section of the data message is generated correctly. The time stamp section is common to devices and connection types.

<b>Requirement Number</b>	<b>Requirement</b>
FRS45	The Base format Time Stamp CRC calculation shall include: Producer Identifier byte (PID) (by CRC-S1) The (Mode Byte) AND (0x1F) (by CRC-S1) Time Stamp byte(s) (by CRC-S1)

TST17 The Base Format Producer Packet Time CRC Positive test shall confirm that the producer is generating the Time Stamp CRC correctly.

##### **Required Initial Conditions:**

1. A safety I/O connection must exist between the test and the Producer DUT. The connection process has previously exchanged the PID and CID between the originator and the target.
2. A data message is received by the test.



**Test Procedure:**

1. The time stamp section of the data message is verified to be built correctly.
2. The test calculates the time stamp CRC using the received time stamp value. The calculated time stamp CRC should be equal to the received time stamp CRC

**F-3.5.1.5 TST18 - Producer Time Correction CRC**

The Time Correction section consists of the Mcast\_Byte, time correction value, Mcast\_Byte\_2, and CRC. This positive multi-cast producer test will test both EF and base format time correction packets. This test will also verify the Time Correction packet is concatenated to the data packet when the connection is on EtherNet/IP.

Requirement Number	Requirement
FRS30	For devices on a non-DeviceNet network, the data and Time Correction_Value shall be concatenated and sent to the consumers via a single multi cast transport connection
FRS68	The Time Correction CRC calculation shall include: Producer Identifier (PID) (by CRC-S3) Mcast_Byte (by CRC-S3) Time_Correction_Value (by CRC-S3)
FRS69	Mcast_Byte_2 of the Time Correction message shall not be included in the Safety CRC
FRS371	When a connection is established with the Extended format, the EF Time Correction format shall be used, otherwise the base format shall be used

TST18 The Producer Time Correction CRC Positive Test shall confirm that the producer is generating the Time Correction message correctly.

**Required Initial Conditions:**

1. This test will function for both the Extended Format and the base format

**Test Procedure:**

1. Establish a safety I/O connection of either the EF or Base format with the Producer DUT. The connection will exchange the PID and if necessary the rollover count between the originator and the target.
2. If the connection is on EtherNet/IP, verify the Time Correction message is concatenated to the data packet with a "NULL" Time Correction packet.
3. At the first Ping Request, send a Time Coordination response of the proper format.
4. If the connection is on EtherNet/IP, verify a valid, non-NULL Time Correction message with the proper consumer number has been concatenated to the produced data packet.
5. The test calculates the Time Correction CRC using the PID parameters received during connection establishment.
6. The calculated time correction CRC should be equal to the received time correction CRC.
7. If the base format is being used, the test calculates the time correction CRC with the Mcast\_Byte\_2 included. The calculated time correction CRC with the Mcast\_Byte\_2 included should not be equal to the received time correction CRC.



**F-3.5.1.6 TST19 - Producer Time Correction Mcast Byte & Mcast Byte 2**

The MCast\_Byte and MCast\_Byte\_2 fixed parameters are tested. The Parity\_Even bit (bit7) forms even parity on bits 0-6. The time correction reserved bits are set to 0. The Mcast\_Byte\_2 is derived from the Mcast\_Byte with fixed rules.

Requirement Number	Requirement
FRS62	Parity_Even in the Time Correction message shall be the even parity of bits 0 through 6 of the Message
FRS66	The following rule shall be used for the calculation of the parity bit in the Time Correction message: Bit 7 of the MCast_Byte form even parity on the MCast_Byte.
FRS67	The following rule shall be used for setting of the bits in Mcast_Byte_2 in the Time Correction message: $\text{MCast\_Byte\_2} = ((\text{MCast\_Byte} \text{ XOR } 0\text{xFF}) \text{ AND } 0\text{x55}) \text{ OR } (\text{MCast\_Byte} \text{ AND } 0\text{xAA})$
FRS215	The Time Correction Reserved bits shall be set to 0 by the producer

TST19 The Base format Producer Time Correction Mcast\_Byte & Mcast\_Byte\_2 test shall confirm that the producer is setting the parity, Mcast and reserved bits correctly in the Time Correction Message.

**Required Initial Conditions:**

1. A safety I/O connection must exist between the test and the Producer DUT. The connection process has previously exchanged the PID and CID between the originator and the target.
2. A Time Correction message has been received by the test

**Test Procedure:**

1. Bit 6 of the Mcast\_Byte is verified to be 0.
2. Bit 6 of the Mcast\_Byte\_2 is verified to be 1.
3. The test calculates the parity for bits 0-7 of the Mcast\_Byte.
4. The parity calculation should be equal to 0.
5. The received Mcast\_Byte is used to generate the Mcast\_Byte\_2.
6. The generated Mcast\_Byte\_2 should be equal to the received Mcast\_Byte\_2.

**F-3.5.1.7 TST111 – Extended Format Producer Time Correction Mcast Byte**

The MCast\_Byte parameter is tested in the Extended Format Time Correction Message. The Parity\_Even bit (bit7) forms even parity on bits 0-6. The time correction reserved bits are set to 0.

Requirement Number	Requirement
FRS62	Parity_Even in the Time Correction message shall be the even parity of bits 0 through 6 of the Message
FRS66	The following rule shall be used for the calculation of the parity bit in the Time Correction message: Bit 7 of the MCast_Byte form even parity on the MCast_Byte.
FRS215	The Time Correction Reserved bits shall be set to 0 by the producer
FRS371	When a connection is established with the Extended format, the EF Time Correction format shall be used; otherwise the base format shall be used



TST111 The Extended Format Producer Time Correction Mcast\_Byte test shall confirm that the producer is setting the parity, Mcast and reserved bits correctly in the Time Correction Message.

**Required Initial Conditions:**

1. An EF safety I/O connection must exist between the test and the Producer DUT. The connection process has previously exchanged the PID and CID between the originator and the target.
2. A Time Correction message has been received by the test

**Test Procedure:**

1. Bit 6 of the Mcast\_Byte is verified to be 0.
2. The test calculates the parity for bits 0-7 of the Mcast\_Byte.
3. The parity calculation should be equal to 0.

**F-3.5.1.8 TST20 – Base Format Producer Single-Cast**

Requirement Number	Requirement
FRS16	For single-cast safety connections, the Single-cast SafetyValidatorClient shall produce safety data to the Single-cast SafetyValidatorServer as defined by the Expected Packet Interval (EPI) rate.
FRS18	The Single-cast safety data producer shall use the time value returned in the ping response (Time Coordination message) to adjust the time stamp sent the Single-cast produced data.
FRS89	At each single-cast producer ping interval, the ping count shall be incremented in the next available safety message header.
FRS90	The single-cast producer ping interval shall be a multiple of the EPI; ping requests shall be made at rates based on the Ping_Interval_EPI_Multiplier
FRS241	If the connection type is single-cast, the Max_Consumer_Number shall be equal to 1.
FRS281	For Single-Cast, the flag Init_Complete_Out shall indicate that the first time coordination message has been received by the producer and the producer is producing data with the time stamp relative to the consumer's clock.
FRS283	For Single-Cast, the Data_Time_Stamp value shall be set to 0 by the producer until the first time coordination section is received

TST20 The Base format Single-Cast Data Production test shall establish a single-cast connection to the producing DUT and confirm the data is produced properly

**Required Initial Conditions:**

1. An base format single-cast connection must exist between the test and the Producer DUT. The connection process has previously exchanged the PID and CID between the originator and the target.
2. The 1st single-cast message has been received by the test

**Test Procedure:**

1. The test verifies that the time value is 0 in the initial data message.
2. The test verifies that the Data Time Stamp is = 0
3. The initial ping count value is saved.



4. The test verifies that the max consumer number = 1
5. The test responds with a valid time coordination message.
6. The 2<sup>nd</sup> data message is received.
7. The test verifies that the data message was received at the EPI rate.
8. The test verifies that the time value is a value that is relative to the time coordination message's consumer time value.
9. As additional data messages are received, steps 7, 8, and 9 are repeated for each message.
10. The test will keep a count of the data messages received. The ping count value is monitored and when a change is detected the test will compare the message count to the ping count interval value. This test will be executed for a period of time.

**F-3.5.1.9 TST121 – Extended Format Producer Single-Cast**

Requirement Number	Requirement
FRS16	For single-cast safety connections, the Single-cast SafetyValidatorClient shall produce safety data to the Single-cast SafetyValidatorServer as defined by the Expected Packet Interval (EPI) rate.
FRS18	The Single-cast safety data producer shall use the time value returned in the ping response (Time Coordination message) to adjust the time stamp sent the Single-cast produced data.
FRS89	At each single-cast producer ping interval, the ping count shall be incremented in the next available safety message header.
FRS90	The single-cast producer ping interval shall be a multiple of the EPI, ping requests shall be made at rates based on the Ping_Interval_EPI_Multiplier
FRS241	If the connection type is single-cast, the Max_Consumer Number shall be equal to 1.
FRS281	For Single-Cast, the flag Init_Complete_Out shall indicate that the first time coordination message has been received by the producer and the producer is producing data with the time stamp relative to the consumer's clock
FRS283	For Single Cast, the Data_Time_Stamp value shall be set to 0 by the producer until the first time coordination section is received
FRS376	The rollover count used in the Extended Format Single-Cast CRC shall be zero until the initial Time Coordination exchange has been completed

TST121 The Extended Format Single-Cast Data Production test shall establish a single-cast connection to the producing DUT and confirm the data is produced properly

**Required Initial Conditions:**

**Test Procedure:**

1. Originate an Extended Format single-cast connection with the Producer DUT and provide the Initial Time Stamp and Initial Rollover Value. The connection process will also exchange the PID & CID between the originator and the target.
2. The 1st single-cast message has been received by the test
3. The test verifies the CRC calculates correctly with a 0 Rollover Value.
4. The test verifies that the data time stamp = 0
5. The initial ping count value is saved.
6. The test verifies that the max consumer number = 1



7. The test responds with a valid Extended Format time coordination message.
8. The 2<sup>nd</sup> data message is received.
9. The test verifies that the data message was received at the EPI rate.
10. The test confirms the CRC calculates correctly with the proper Rollover Count Value included
11. The test verifies that the data time stamp value is a value that is relative to the time coordination message's consumer time value
12. As additional data messages are received, steps 7, 8, and 9 are repeated for each message.
13. The test will keep a count of the data messages received. The ping count value is monitored and when a change is detected the test will compare the message count to the ping count interval value. This test will be executed for a period of time.
14. The messages will continue until a Time Stamp rollover occurs.
15. The test will confirm that the next message CRC calculates correctly with an updated Rollover Count Value.

#### **F-3.5.1.10 TST21 - Producer Run/Idle Usage**

The Run\_Idle bit is used to indicate the state of the data in the data message. This functionality is common to single-cast and multi-cast.

<b>Requirement Number</b>	<b>Requirement</b>
FRS104	Once the producer has correctly verified a successful response to a forward open message, the producer shall start producing safety data at the periodic rate but with the run/idle bit set to idle (single-cast) or the Consumer Active Idle set to Idle (multi-cast).
FRS106	The producer shall maintain the run/idle or Consumer_Active_Idle bit idle until the initial Time Coordination sequence is successfully completed.
FRS179	The Run_Idle bit value of 0 shall indicate Idle, and the Run_Idle bit value of 1 shall indicate Run
FRS180	For single-cast connections, the Run_Idle bit shall be set to Idle if the producing application is Idle or if the producer is waiting for Time Coordination Message information to be received at the initiation of data production
FRS181	If Time Coordination Message information has been received; the Run_Idle bit shall be set to the Run_Idle status indicated by the producing application
FRS183	The N_Run_Idle complement bit shall always be the complement of the Run_Idle bit
FRS218	If the producing application is actively controlling data the Run_Idle indication shall indicate Run. If the producing application is not actively controlling the data and the data should be put in the safety state, the Run_Idle indication shall indicate Idle.

TST21 The Producer Run/Idle Usage Test shall confirm that the producer is controlling the Run/Idle indication properly .

#### **Required Initial Conditions:**

1. This test will function correctly for both base format and Extended Format messages
2. A safety I/O connection must exist between the test and the Producer DUT. The connection process has previously exchanged the PID and CID between the originator and the target.
3. The 1<sup>st</sup> single-cast or Multi-cast message has been received by the test



**Test Procedure:**

1. If the connection type is single-cast verify that the Run\_Idle bit is set to 0.
2. If the connection type is multi-cast, verify that the Run\_Idle bit is set to the Application state,
3. The test will send a valid time coordination message.
4. The DUT will validate the time coordination message and set the Run\_Idle bit **to the application state**.
5. The test will receive the data message and verify that the Run\_Idle bit is set to **the Application state**.
6. If the connection type is multi-cast, the test will receive a Time Correction Packet
7. Verify that the multicast active idle bit is set

**F-3.5.1.11 TST22 - Producer Ping Count Usage**

The Ping\_Count bits are used to request time coordination messages. A time coordination message is expected to be received after every ping\_count change.

Requirement Number	Requirement
FRS70	Upon connection establishment, the producer shall request an initial ping response.
FRS75	Once each Ping Interval, the producer shall request another ping response from the consumer.
FRS232	The Ping_Count_Interval shall be equal to the EPI times the Ping_Interval_EPI_Multiplier.
FRS233	The Ping_Count_Interval shall define the rate at which Time Coordination Messages will be requested and sent.
FRS235	A legal Ping_Interval_EPI_Multiplier shall have a minimum value determined from the equation $\text{Max\_Consumer\_Number} * \text{Timeout\_Multiplier.PI} + 15$ , and a maximum less than 1000. The default values shall be 100 for Multi-cast and 19 for Single Cast
FRS239	The Ping_Interval_EPI_Multiplier shall be greater than or equal to: $\lceil \{ \text{the Max Timeout\_Multiplier.PI(C\#) for all consumers} \} * \text{Max\_Consumer\_Num} \rceil + 15$
FRS318	The Ping_Interval_EPI_Multiplier shall be set small enough to ensure that the Ping_Count_Interval is less than or equal to 100 seconds.
FRS333	Producers shall allow Time Coordination responses to arrive during the Ping interval or $\text{Ping\_Interval} + 1$

TST22 The Producer Ping Count Usage positive test shall confirm the producer DUT is generating Ping Count changes properly.

**Required Initial Conditions:**

1. This test will support both base and Extended Format connections.
2. A safety I/O connection must exist between the test and the Producer DUT. The connection process has previously exchanged the PID and CID between the originator and the target.
3. A single-cast or Multi-cast message has been received by the test

**Test Procedure:**

1. In response to the 1st Ping Request, send a Time Coordination Response in  $\text{Ping\_Interval} + 1$



2. Confirm the Producer DUT does not close the connection
3. The DUT sends a data message with the Ping\_Count bits set to a value that will trigger a time coordination message to be sent.
4. The test receives the data message and verifies the ping\_count. The test will send a valid time coordination message.
5. The DUT will continue to send data messages and send a ping count request at the ping count interval.
6. The test will respond to ping count changes during the Ping\_Interval. The test will monitor the time between ping count changes and compare the time to the ping count interval. The test received the Ping\_Interval EPI Multiplier and the EPI values via the forward open. The test will calculate the ping count interval that is used for the comparison.
7. The DUT will be reconfigured so that various EPI and Ping\_Interval\_EPI\_Multiplier values can be tested. After each configuration steps 1 through 4 will be executed.
8. The configurations will be chosen so that the Ping\_Interval\_EPI\_Multiplier range will be tested.
9. The configurations will be chosen so that the Ping\_Count\_Interval range will be tested

#### **F-3.5.1.12 TST23 – Base Format Multi-Cast Production to a Single Consumer**

The requirements that are unique to the multi-cast connection type but not specific to multiple consumers are verified in this test with the tester simulating 1 consumer.

Requirement Number	Requirement
FRS24	The produced Multi-Cast safety data shall include a Producer Time Stamp with each safety data production.
FRS26	The Multi-cast Consumer_Time_Values shall be used by the Multi-cast safety data producer to generate a Time_Correction_Value for each safety data consumer
FRS27	A Time_Correction_Value shall be sent by the Multi-cast safety data producer to each safety data consumers each Ping Interval
FRS182	For multi-cast connections, the Run Idle bit shall be set to the Run Idle status indicated by the producing application.
FRS204	The Reserved bits shall be ignored by the producer except for CRC and Ack_Byte_2 checking
FRS209	The Multi_Cast_Active_Idle bit value of 0 shall indicate Idle, and the Multi_Cast_Active_Idle bit value of 1 shall indicate Active.
FRS210	Once the Multi_Cast_Active_Idle bit has been set to Active after 1st data production, this bit shall not be set back to idle until the safety connection is re-initialized
FRS234	The Ping_Count_Interval shall define the rate at which Time Correction sections are sent to each consumer.
FRS237	All Time_Correction sections shall be sent out by the client within the Ping_Interval.
FRS285	For Multi-Cast, the Data_Time_Stamp value shall always be set equal to the producers clock.
FRS358	Multi_Cast_Active_Idle in the Time Correction message shall indicate Idle if transmitted before this consumer has responded with a valid time coordination message

**TST23** The Base Format Producer Multi-cast positive test shall confirm that the producer DUT behaves correctly when connected to 1 consumer.



**Required Initial Conditions:**

**Test Procedure:**

1. Setup a safety I/O connection of a type supported by the DUT.
2. The test will verify that the Time Correction message is not sent prior to the Time Coordination.
3. The test will send a valid time coordination message.
4. The test will verify that the time stamp is valid
5. The DUT will send a 2nd data message.
6. The test will verify that the 2nd data message contains a valid time\_correction value based on the consumer\_time\_value
7. The test will verify that the Multi\_Cast\_Active\_Idle bit is set to 1
8. The test will verify that the time stamp is valid.
9. The test will verify that the time\_correction value is using the consumer\_time\_value.
10. The test will send a valid time coordination message.
11. After the initialization process the time\_correction data is sent at the ping\_interval.
12. The test will keep a count of the data messages received. The ping count value is monitored and when a change is detected the test will compare the message count to the ping count interval value. The test will verify the time\_correction value. This test will be executed for a period of time.

**F-3.5.1.13 TST122 – Extended Format Multi-Cast Production to a Single Consumer**

There are some special requirements for Extended Format Multi-cast connections that aren't checked by TST23. This tests duplicates all the common tests from TST23 and adds the special steps for the Extended Format.

Requirement Number	Requirement
FRS24	The produced Multi Cast safety data shall include a Producer Time Stamp with each safety data production.
FRS26	The Multi-cast Consumer_Time_Values shall be used by the Multi-cast safety data producer to generate a Time_Correction_Value for each safety data consumer
FRS27	A Time_Correction_Value shall be sent by the Multi-cast safety data producer to each safety data consumers each Ping Interval.
FRS182	For multi-cast connections, the Run_Idle bit shall be set to the Run_Idle status indicated by the producing application.
FRS204	The Reserved bits shall be ignored by the producer except for CRC and Ack_Byte_2 checking.
FRS209	The Multi_Cast_Active_Idle bit value of 0 shall indicate Idle, and the Multi_Cast_Active_Idle bit value of 1 shall indicate Active.
FRS210	Once the Multi_Cast_Active_Idle bit has been set to Active after 1st data production, this bit shall not be set back to idle until the safety connection is re-initialized
FRS234	The Ping_Count_Interval shall define the rate at which Time Correction sections are sent to each consumer.
FRS237	All Time_Correction sections shall be sent out by the client within the Ping_Interval.
FRS285	For Multi-Cast, the Data_Time_Stamp value shall always be set equal to the producers clock.



Requirement Number	Requirement
FRS358	Multi_Cast_Active_Idle in the Time Correction message shall indicate Idle if transmitted before this consumer has responded with a valid time coordination message
FRS377	When Extended Format multi-cast connections are originated by the consumer (i.e. multi-cast inputs); the target producer shall provide initialization values in the SafetyOpen Response.
FRS378	The active seeding of the CRC with the rollover count in Extended Format Multi-cast messages shall begin immediately on first production.

TST122 The Extended Format Producer Multi-cast positive test shall confirm that the producer DUT behaves correctly when connected to 1 consumer.

**Required Initial Conditions:**

**Test Procedure:**

1. Setup an Extended Format safety I/O connection of a type supported by the DUT. The test will capture the Initial Time Stamp and Initial Rollover value in the SafetyOpen or Safety Forward Open response
2. The test will verify that the CRC of data packet calculates correctly with the Initial Rollover Count value
3. The test will verify that the Extended Format Time Correction message is not sent prior to the Time Coordination.
4. The test will send a valid Extended Format time coordination message.
5. The test will verify that the time stamp is valid
6. The DUT will send a 2nd data message.
7. The test will verify that the 2nd data message contains a valid time\_correction value based on the consumer time value
8. The test will verify that the Multi\_Cast\_Active\_Idle bit is set to 1
9. The test will verify that the time stamp is valid.
10. The test will verify that the time\_correction value is using the consumer\_time\_value.
11. The test will send a valid time coordination message.
12. After the initialization process the time\_correction data is sent at the ping\_interval.
13. The test will keep a count of the data messages received. The ping count value is monitored and when a change is detected the test will compare the message count to the ping count interval value.
14. The test will verify the time\_correction value.
15. This test will be executed until a Time Stamp Rollover occurs.
16. The test will verify on the test data packet that the CRC calculates correctly with the updated Rollover count.



### **F-3.5.1.14 TST24 - Multi-Cast Production to Multiple Consumers**

The requirements that are unique to the multi-cast connection type and are specific to multiple consumers are verified in this test. The tester will simulate multiple consumers.

<b>Requirement Number</b>	<b>Requirement</b>
FRS23	For multi-cast safety connections the SafetyValidatorClient shall produce Safety Data to multiple (n = 2 to 15) SafetyValidatorServers as defined by the Expected Packet Interval (EPI) rate.
FRS60	The time correction message shall be sent from a multi-cast producer to each multi-cast consumer, once every ping interval.
FRS65	The Consumer # in the Time Correction message shall indicates the number of the consumer that this message is directed to.
FRS95	Time correction messages shall be sent to the multi-cast consumers at a rate to ensure that each consumer gets a time correction message within a single producer ping interval
FRS96	For Non-DeviceNet networks the time correction message shall be concatenated with the safety data message and sent as a single message
FRS205	For multi-cast produced data the Multi_Cast Active Idle, and Consumer_Time_Correction_Value shall be directed to the consumer indicated by the Consumer # field of the message.
FRS242	For multi-cast connections, the legal range of Max_Consumer_Number values shall be from 1 through 15.
FRS257	For multi-cast producers, Time Correction messages shall be sent to the n consumers (where n = 1 thru Max_Consumer Number), during the 8th thru n+8 EPI's of the Ping Interval.
FRS258	The Time Correction message of consumer 1 shall be sent first, followed by 2, 3,...,n.
FRS358	Multi_Cast_Active_Idle in the Time Correction message shall indicate Idle if transmitted before this consumer has responded with a valid time coordination message

TST24 The Multi-cast Producer to Multiple Consumer test shall confirm the requirement related to maintaining connections to multiple consumers.

#### **Required Initial Conditions:**

1. This test will function for base format and Extended Format messages
2. The DUT should be configured to support its maximum number of consumers "if necessary".

#### **Test Procedure:**

1. The test will open DUT's maximum # of connections one at a time.
2. The test will emulate the DUT maximum # of consumers and will control when a consumer becomes active.
3. The test will execute the following steps for each consumer as it becomes active.
4. The DUT and test open a connection.
5. The DUT produces a valid data message.
6. The test receives the data message and responses with a valid time coordination message.
7. The DUT produces the initial time correction message with the correct consumer number. The Multi\_Cast\_Active\_Idle bit is set to 1 indicating that the consumer's time coordination message was valid.



8. The test will verify the data message production at the EPI rate.
9. The test will verify that the time correction message is received once every ping interval.
10. The test will verify that the time correction messages were sent starting at the 9th production of the ping interval. Consumer #1 at the 9th, consumer #2 at the 10th,..., Consumer MAX at the (MAX+9)th production of the ping interval.
11. The test will verify that time correction messages were sent in increasing, sequential order from 1 to the maximum consumer number.

### **F-3.5.2 Positive Consumer Tests**

When testing a DUT consumer the tester will be the producer. The tester will generate data and Time Correction messages, and verify any Time Coordination message received. These tests will be used by both the originator and target positive test engines.

#### **F-3.5.2.1 TST25 – Single-cast Consumer Time Coordination Message Generation**

The consumer must generate the time coordination message in response to a ping count change to maintain a connection. The tests in this section will verify a consumer DUT is properly generating the sections of the time coordination message. These requirements are common to devices and connection types.

<b>Requirement Number</b>	<b>Requirement</b>
FRS17	The Single-cast SafetyValidatorServer shall respond to a ping request and produces a Time Coordination message with a Time_Value to the Single-cast SafetyValidatorClient that is used by the safety data producer.
FRS48	The time coordination section shall be sent from the consumer to the producer in response to a change in the ping count.
FRS52	The Parity Even bit in the Time Coordination message shall be even parity of bits 0 through 6 of the message
FRS53	The Ping_Response_Bit in the Time Coordination message shall indicate that a ping response is being sent
FRS55	Ping_Count_Reply in the Time Coordination message shall indicates which ping request the consumer is responding to
FRS56	The following rule shall be used for the calculation of the parity bit in the Time Coordination message: Bit 7 of the Ack_Byte form even parity on the Ack_Byte.
FRS57	The following rule shall be used for setting of the bits in Ack_Byte_2 of the Time Coordination message: $Ack\_Byte\_2 = ((Ack\_Byte \text{ XOR } 0xFF) \text{ AND } 0x55) \text{ OR } (Ack\_Byte \text{ AND } 0xAA)$
FRS58	The Time Coordination CRC calculation shall include. Consumer Identifier (CID) (by CRC-S3 or CRC-S5), Ack Byte (by CRC-S3 or CRC-S5), Consumer Time Value (by CRC-S3 or CRC-S5)
FRS59	Ack_Byte_2 in the Base Time Coordination message shall not be included in the Safety CRC
FRS71	When the consumer sees a change in the ping count, the consumer shall prepare a Time Coordination response message that contains the consumer's value of its clock count
FRS142	The SafetyValidatorServer shall perform Ping_Count checking on every message
FRS203	The Time Coordination Reserved bits shall be set to 0 by the consumer.
FRS291	A consumer of safety data shall be able to detect when the Ping_Count has changed



Requirement Number	Requirement
FRS332	If a SafetyValidatorServer detects that the ping count has changed, it shall produce a time coordination message within that Ping Interval or within 5 seconds, whichever is less.
FRS370	When a connection is established with the Extended format, the EF Time Coordination format shall be used; otherwise the base format shall be used.

TST25 The Time Coordination generation test is a positive test which shall confirm that the Time Coordination messages are generated properly by the consumer DUT.

**Required Initial Conditions:**

1. This test will function for both the EF and base format connections
2. Set up a safety I/O connection with the DUT
3. The connection process exchanged the PID and CID between the originator and the target.

**Test Procedure:**

1. Send a data message with a ping count change
2. The test will confirm a Time Coordination response is sent. The connection is fully established.
3. Confirm the Parity bit is correct for even parity in bit 7 of Ack\_Byte
4. If the base format is being used, confirm that Ack\_Byte\_2 algorithm generated the correct value
5. Confirm that the Ping count is equal to the ping count value sent
6. Confirm that the Reserved bits 4, 5, & 6 are set to 0
7. If the base format is being used, confirm the CRC includes the CID and CID excludes the Ack\_Byte 2

**F-3.5.2.2 TST26 – Consumer Positive CRC/PID/CID Test**

A number of requirements are confirmed by simply communicating with the DUT without errors.

The Producer ID (PID) is the initial seed value for the safety CRC's of all messages from the data producer of a connection. The Consumer ID (CID) is the initial seed value for the safety CRC's of all messages from the consumer of a connection. Every message that is transmitted or received will have the safety CRC verified.

Requirement Number	Requirement
FRS32	All safety messages and safety response messages shall be encoded with the safety CRC's defined in Appendix E
FRS142	The SafetyValidatorServer shall perform Ping_Count checking on every message
FRS155	The Producer Identifier (PID) shall be used as an initial seed value in the CRCs in the runtime protocol.
FRS156	The Consumer shall obtain the Producer Identifier from either the SafetyOpen or SafetyOpen response (depending upon originator) and use the value as an initial seed in runtime checking the safety CRCs.



<b>Requirement Number</b>	<b>Requirement</b>
FRS291	A consumer of safety data shall be able to detect when the Ping_Count has changed.
FRS330	The Consumer Identifier (CID) shall be used as an initial seed value in the CRC in the Time Coordination message
SRS152	During connection establishment the producing and consuming nodes shall exchange their respective PID/CID information (Vendor ID + Device Serial Number + Connection Serial Number). The originator sends its ID in the SafetyOpen request, and the target returns its ID as part of the SafetyOpen Success response.
SRS178	The PID shall be used to seed Data production CRCs and Time Correction CRC, and the CID shall be used to seed the Time Coordination CRC.

**TST26** The Consumer Positive Run-time Test shall be a positive test to confirm that the consumer DUT is both creating proper safety CRCs with the CID and interpreting valid safety CRCs with PID.

**Required Initial Conditions:**

1. This test will function for both the EF and base format connections
2. A connection must exist between the tester and the DUT. The connection process has previously exchanged the PID/CID from the originator to the target.

**Test Procedure:**

1. All messages received from the DUT will have the safety CRC's recalculated with the CID and the received data.
2. The safety CRC's will be verified to be equal.
3. All messages transmitted to the DUT will have the safety CRC's calculated with the PID by the test.

This test will not generate CRC error conditions or confirm such detection at the DUT. While this test is running, no error or faults should be generated by the DUT

**F-3.5.2.3 TST27 – Multi-cast Consumer Time Coordination Message Generation Test**

The consumer must generate the time coordination message in response to a ping count change within specific time windows in the Ping Interval. The tests in this section will verify a consumer DUT is properly generating the sections of the time coordination message and at the proper time.

<b>Requirement Number</b>	<b>Requirement</b>
FRS25	The Multi-Cast SafetyValidatorServers shall respond to a ping request by sending a Time Coordination message with a Consumer Time Value to the SafetyValidatorClient.
FRS94	All mult.-cast consumers shall respond to a change in the ping request value, consumers other than consumer 1 shall wait Consumer Number -1 times the EPI to reply.
FRS148	Multicast-consumers connecting to an existing producer shall send the Time Coordination message immediately upon the first valid reception.
FRS332	If a SafetyValidatorServer detects that the ping count has changed, it shall produce a time coordination message within that Ping Interval or within 5 seconds, whichever is less.



TST27 The Multi-cast Time Coordination Spread Test shall confirm the DUT multi-cast consumer will properly respond in an EPI slot consistent with its assigned consumer #

**Required Initial Conditions:**

1. This test will function for both the EF and base format connections

**Test Procedure:**

1. Set up the DUT to Establish a safety I/O connection with the tester
2. Assign it an initial consumer # of 1
3. Trigger a ping count change
4. Run TST25 – Single-cast Consumer Time Coordination Message Generation
5. Run TST26 – Consumer Positive CRC/PID/CID Test
6. Confirm a Time coordination response is generated immediately
7. Continue communication at the next ping interval generate another ping count change
8. Confirm the Time Coordination response is generated in the correct time slot
9. Stop producing data long enough for the DUT to close the connection and attempt to re-establish
10. Repeat from step 1 with consecutively increasing consumer # assignments from 1 – 15
11. Confirm that the ping responses are initially immediate and then subsequently delayed to Consumer # - 1 data productions

**F-3.6 Common Negative Consumer Tests**

This section will define behavior that is common to all devices regardless of what subset of Safety Functionality is implemented. In other words, all safety devices must pass the tests described in this section.

**F-3.6.1 TST28 – Base Format Incorrect Sequence/Insertion Detection**

Requirement Number	Requirement
FRS84	Consumers using base format connections shall see the Timestamp increase for each successive period or the connection shall be terminated
FRS299	The Last Data_Time_Stamp shall be used to detect out of sequence messages

TST28 The Base Format Incorrect Sequence/Insertion Detection test shall confirm that the DUT detects non-consecutive Time Stamps and treats these as errors.

1. Set up a safety I/O connection of a type supported by the DUT
2. The establishment of a safety I/O connection is a positive test case for the Time\_Stamp generation.
3. The test will calculate the data age for each packet to verify the Time\_Stamp correctness.
4. The connection will run for a set period of time.
5. The Time\_Stamp register/counter is a cyclic counter and will rollover. At the rollover condition the most recent Time\_Stamp will be less than the previous Time\_Stamp. The data age check will verify this rollover condition.
6. Send a Time\_Stamp value that is less than the current Time\_Stamp value.



7. Verify that the connection is terminated.
8. The test will define expected fail-safe behavior

### **F-3.6.2 TST112 – Extended Format Incorrect Sequence/Insertion Detection**

<b>Requirement Number</b>	<b>Requirement</b>
FRS372	Consumers using Extended Format connections shall see the Timestamp increase for each successive period. If the consumer receives a packet with an out-of-sequence Timestamp, it shall drop the packet unless the Consumer Fault Counter has reached the Max Fault Number; in which case the connection shall be terminated.
FRS299	The Last_Data_Time_Stamp shall be used to detect out of sequence messages
FR375	When the connection is a EF Consuming connection, the Producer/Consumer Fault Counter attribute shall reflect the Consumer_Fault_Counter. When the connection is a EF Producing connection, the attribute shall reflect the Producer_Fault_Counter.

TST112 The Extended Format Incorrect Sequence/Insertion Detection test shall confirm that the DUT detects non-consecutive Time Stamps and drops the packet until the Max Fault Count is reached, then terminates the connection.

**The following procedure will be followed:**

1. Set up a EF safety I/O connection of a type supported by the DUT with a Max\_Fault\_Number = 5.
2. The establishment of a safety I/O connection is a positive test case for the Time\_Stamp generation.
3. The test will calculate the data age for each packet to verify the Time\_Stamp correctness.
4. The connection will run for a set period of time.
5. The Time\_Stamp register/counter is a cyclic counter and will rollover. At this point, the producer and consumer should have incremented their rollover counter. At the rollover condition the most recent Time\_Stamp will be less than the previous Time\_Stamp.
6. Send a Time\_Stamp value that is less than the current Time\_Stamp value for (Max\_Fault\_Number – 1) times.
7. Verify that the packet is dropped by inspecting the Producer/Consumer Fault Count attribute of the Safety Validator, but the connection is maintained.
8. Send one more Time\_Stamp values less than the current Time\_Stamp
9. Verify that the connection is terminated

### **F-3.6.3 TST29 - Message Corruption Detection**

<b>Requirement Number</b>	<b>Requirement</b>
FRS5	A message corruption shall be detected when the data and CRC-Sx are calculated and compared. This error shall cause the connection to be: Terminated IF (Base format) OR (Extended Format AND Consumer Fault Count) > Max_Fault_Number OR Dropped IF Extended Format AND Consumer Fault_Count < Max_Fault_Number..
FRS6	A corrupted message that was not detected by the link or CRC-Sx check (i.e., error that exceeded the Hamming distance of CRC) shall be detected when the actual and complemented copies of the data are compared in the consumer.



Requirement Number	Requirement
FRS86	If the transmitted CRC-Sx and data are checked and any found to be in error or the data cross-check is found to be in error, the consumer shall either drop the packet (Extended Format AND Consumer_Fault_Count < Max_Fault_Count), or terminate the connection and go to the defined safety state (Base format or Extended Format AND Consumer_Fault_Count ≥ Max_Fault_Count).
FRS375	When the connection is a EF Consuming connection, the Producer/Consumer Fault Counter attribute shall reflect the Consumer Fault Counter. When the connection is a EF Producing connection, the attribute shall reflect the Producer_Fault_Counter

TST29 The Message Corruption Detection test shall set up a safety I/O connection of a type supported by the DUT and simulate corruption by setting invalid Safety CRCs.

**Initial conditions:**

1. This test will support both the EF and base format connections
2. Run SPTE as a target

**The following procedure will be followed:**

1. Establish a standard format connection (if supported)
2. Send a data packet with a corrupt Data CRC
3. Verify that the connection is terminated.
4. Repeat the procedure for the complement data CRC, the Time Stamp CRC, Time Correction and the complemented data.
5. If the DUT supports Multi-cast consumer connections, repeat the test with the Time Correction message
6. Establish a High-Availability format connection (if supported)
7. Send a data packet with a corrupt Data CRC for Max\_Fault\_Count – 1 times
8. Verify that the packet is dropped and connection continued.
9. Send a data packet with a corrupt Data CRC one more time
10. Verify that the connection is terminated
11. Repeat the procedure for the complement data CRC, the Time Stamp CRC, Time Correction and the complemented data.
12. If the DUT supports Multi-cast consumer connections, repeat the test with the Time Correction message

**F-3.6.4 TST30 - Message Delay Detection by Consumers**

Requirement Number	Requirement
FRS3	The safety protocol requires messages to occur within defined time expectations. Messages received later than these time expectations shall be treated as errors, resulting in the connection's termination.
FRS74	The consumer shall use the value received from the producer to calculate the age of the data by subtracting the producer's time stamp from its current clock count. If the calculated delta is less than the maximum delta specified by the user, the data is ok and the data is used by the application within the device. If the delta is greater than the user specified value, the data is deemed to be too old to use and the connection is terminated.
FRS80	All consumers shall check the age of the data received by checking the time stamps



Requirement Number	Requirement
	received.
FRS87	If a message is delayed such that it is received 1. beyond the consumer activity monitor (link-triggered application) or 2. the data age exceeds the network time expectation, then all further messages are ignored, and the connection shall be terminated.

TST30 The Message Delay Detection Test shall set up a safety consumer connections of all types supported by the DUT and send data at a rate that exceeds the agreed upon EPI.

**Initial conditions:**

1. This test will support both the EF and base format connection

**The following procedure will be followed:**

1. Set up a safety I/O connection of a type supported by the DUT
2. The establishment of a safety I/O connection is a positive test case for the message timing.
3. The connection will run for a set period of time.
4. Begin delaying the packet delivery until it exceeds the period set by the Network Time Expectation Multiplier.
5. Verify that the connection is terminated at a time less than or equal to the Network Time Expectation time.

### **F-3.7 Common Negative Producer Tests**

#### **F-3.7.1 TST31 – Base Format Producer Time Coordination Response Failure**

Requirement Number	Requirement
FRS10	When a producer detects an error that requires connection termination it shall terminate the connection, and notify the application of the action
FRS52	The Parity_Even bit in the Time Coordination message shall be even parity of bits 0 through 6 of the message
FRS53	The Ping_Response_Bit in the Time Coordination message shall indicates that a ping response is being sent
FRS55	Ping_Count_Reply in the Time Coordination message shall indicates which ping request the consumer is responding to
FRS56	The following rule shall be used for the calculation of the parity bit in the Time Coordination message: - Bit 7 of the Ack_Byte form even parity on the Ack_Byte.
FRS57	The following rule shall be used for setting of the bits in Ack_Byte_2 of the Time Coordination message: $Ack\_Byte\_2 = ((Ack\_Byte \text{ XOR } 0xFF) \text{ AND } 0x55) \text{ OR } (Ack\_Byte \text{ AND } 0xAA)$
FRS58	The Time Coordination CRC calculation shall include: Consumer Identifier (CID) (by CRC-S3 or CRC-S5) Ack Byte (by CRC-S3 or CRC-S5) Consumer_Time_Value (by CRC-S3 or CRC-S5)
FRS59	Ack_Byte_2 in the Time Coordination message shall not be included in the Safety CRC
FRS98	If a multi-cast time coordination message is lost or delayed past the $(Timeout \text{ Multiplier.PI} + 2)$ ping intervals, an error shall be generated and the connection closed



Requirement Number	Requirement
FRS195	If the Ping_Response bit is 1, it shall indicate that the Consumer_Time_Value, and Ping_Count_Reply, are valid on this production
FRS196	If the Ping_Response bit is 0, it shall indicate that the Consumer_Time_Value, and Ping_Count_Reply, are not valid on this production and should be ignored
FRS229	The Timeout_Multiplier shall be used to determine how many ping intervals (Timeout_Multiplier.PI+2) a producer will wait before faulting a consumer who has failed to send a Time Coordination response to a Ping request
FRS374	For the Base Format, Timeout_Multiplier.PI shall be equal to the Timeout_Multiplier. For the HiAvailabilityFormat, Timeout_Multiplier.PI shall be equal to the smaller of Timeout_Multiplier or 4, whichever is lower.
FRS333	Producers shall allow Time Coordination responses to arrive during the Ping_interval or Ping_interval + 1.

TST31 The Producer Time Coordination Failure Response test shall confirm that the producer DUT performs proper safety-state behavior when communication failures are detected.

**Required Initial Conditions:**

1. Run SPTE as an originator and configure it with a safety I/O connection of a type supported by the DUT

**Test Procedure:**

1. In response to the 1st Ping Request, send a Time Coordination Response in Timeout\_Multiplier.PI + 1 Ping\_Intervals
2. Confirm the Producer DUT does not close the connection
3. In response to the 2<sup>nd</sup> Ping Request, send a Time Coordination Response in Timeout\_Multiplier.PI + 2 Ping\_Intervals
4. Confirm the producer does not closes the connection
5. Ignore all ping requests and send no Time Coordination responses
6. Confirm the producer faults and closes the connection after Timeout\_Multiplier.PI+2 Ping\_Intervals
7. Re-establish the connection
8. Send Time\_Coordination responses with Ping\_Response bit cleared.
9. Confirm the producer faults and closes the connection after Timeout\_Multiplier.PI+2 Ping\_Intervals
10. Re-establish the connection
11. Set the Parity\_Even bit to an incorrect value.
12. The producer will detect the incorrect Time Coordination Message and terminate the connection.
13. Re-establish the connection.
14. Set up a safety I/O connection of a type supported by the DUT
15. Send a valid Time Coordination Message.
16. The producer will calculate the Time Coordination Message using the data received in the data packet. Compare the calculated CRC to the received CRC. The values should be equal.



17. Send an invalid Time Coordination Message. Calculate the CRC using the Ack\_Byte\_2.
18. The producer will calculate the Time Coordination Message using the data received in the data packet including excluding the Ack\_Byte\_2. Compare the calculated CRC to the received CRC. The values should be unequal.
19. The producer will detect the incorrect Time Coordination Message and terminate the connection
20. Re-establish the connection
21. Generate a Time Coordination Message with an incorrect Ping\_Count\_Reply.
22. The producer DUT will detect the incorrect Time Coordination Message with the wrong Ping\_Count\_Reply. The producer will terminate the connection.
23. Re-establish the connection.
24. Generate a Time Coordination Message with an incorrect Ack\_Byte\_2 value.
25. The producer DUT will detect the incorrect Ack\_Byte\_2 value and terminate the connection.
26. Re-establish the connection.
27. Generate a Time Coordination Message with an incorrect Time Coordination Message CRC. The CRC will be calculated without the Connection Originator Identifier.
28. The producer will detect the incorrect Time Coordination Message CRC and terminate the connection.

Because Time Coordination Responses are common to both Multi-cast and Single-cast, this test applies to both I/O types and if a DUT supports both types of Safety producer, the test shall be executed for both.

### **F-3.7.2 TST116 - Extended Format Producer Time Coordination Response Failure**

<b>Requirement Number</b>	<b>Requirement</b>
FRS10	When a producer detects an error that requires connection termination it shall terminate the connection, and notify the application of the action.
FRS52	The Parity_Even bit in the Time Coordination message shall be even parity of bits 0 through 6 of the message
FRS53	The Ping_Response_Bit in the Time Coordination message shall indicates that a ping response is being sent
FRS55	Ping_Count_Reply in the Time Coordination message shall indicates which ping request the consumer is responding to
FRS56	The following rule shall be used for the calculation of the parity bit in the Time Coordination message. - Bit 7 of the Ack_Byte form even parity on the Ack_Byte.
FRS58	The Time Coordination CRC calculation shall include: Consumer Identifier (CID) (by CRC-S3 or CRC-S5) Ack Byte (by CRC-S3 or CRC-S5) Consumer Time Value (by CRC-S3 or CRC-S5)
FRS98	If a multi-cast time coordination message is lost or delayed past the (Timeout Multiplier.PI +2) ping intervals, an error shall be generated and the connection closed
FRS195	If the Ping_Response bit is 1, it shall indicate that the Consumer Time Value, and Ping_Count_Reply, are valid on this production
FRS196	If the Ping_Response bit is 0, it shall indicate that the Consumer_Time_Value, and Ping_Count_Reply, are not valid on this production and should be ignored.



Requirement Number	Requirement
FRS229	The Timeout_Multiplier shall be used to determine how many ping intervals (Timeout_Multiplier.PI+2) a producer will wait before faulting a consumer who has failed to send a Time Coordination response to a Ping request
FRS374	For the Base Format, Timeout_Multiplier.PI shall be equal to the Timeout_Multiplier. For the HtAvailabilityFormat, Timeout_Multiplier.PI shall be equal to the smaller of Timeout_Multiplier or 4, whichever is lower.
FRS333	Producers shall allow Time Coordination responses to arrive during the Ping_interval or Ping_Interval + 1.

TST116 The Extended Format Producer Time Coordination Failure Response test shall confirm that the producer DUT performs proper behavior when packet errors are detected.

**Required Initial Conditions:**

1. Run SPTE as an originator and configure it with a safety I/O connection of a type supported by the DUT

**Test Procedure:**

1. In response to the 1st Ping Request, send a Time Coordination Response in same Ping\_Interval
2. Confirm the Producer DUT does not close the connection
3. In response to the 2<sup>nd</sup> Ping Request, send a Time Coordination Response in the next Ping\_Interval
4. Confirm the producer does not closes the connection
5. Ignore all ping requests and send no Time Coordination responses
6. Confirm the producer faults and closes the connection after Timeout\_Multiplier.PI+2 Ping\_Intervals
7. Re-establish the connection
8. Send Time\_Coordination responses with Ping\_Response bit cleared
9. Confirm the producer faults and closes the connection after Timeout\_Multiplier.PI+2 Ping\_Intervals
10. Re-establish the connection
11. Send Time Coordination responses with an incorrect Parity\_Even bit for (Max\_Fault\_Count) or (Timeout\_Multiplier.PI+2) times; whichever is less
12. Verify that the producer closes the connection after the last transmission but not the prior ones.
13. Re-establish the connection.
14. Send a valid Time Coordination Message.
15. The producer will calculate the Time Coordination Message using the data received in the data packet. Compare the calculated CRC to the received CRC. The values should be equal.
16. Generate a Time Coordination Message with an incorrect Ping\_Count Reply for (Max\_Fault\_Count) or (Timeout\_Multiplier.PI+2) times; whichever is less.
17. Verify that the producer does not close the connection after the last transmission but not the prior ones.



18. Re-establish the connection.
19. Generate a Time Coordination Message with an incorrect Time Coordination Message CRC for (Max\_Fault\_Count) or (Timeout\_Multiplier.PI+2) times; whichever is less.
20. Verify that the producer does not close the connection after the last transmission but not the prior ones.

Because Extended Format Time Coordination messages are common to both Multi-cast and Single-cast, this test applies to both I/O types and if a DUT supports both types of Safety producer, the test shall be executed for both.

### **F-3.7.3 TST32 - Producer Time Coordination No-response**

The Timeout\_Multiplier value is a common parameter for the producer and consumer. This parameter is part of the forward open request. The producer uses it to determine the number Ping\_Intervals it will wait for a time coordination message before a fault occurs. This behavior is common to multi-cast or single-cast connections in both Base and Extended Formats.

Requirement Number	Requirement
FRS229	The Timeout_Multiplier shall be used to determine how many ping intervals (Timeout_Multiplier.PI+2) a producer will wait before faulting a consumer who has failed to send a Time Coordination response to a Ping request
FRS374	For the Base Format, Timeout_Multiplier.PI shall be equal to the Timeout_Multiplier. For the HiAvailabilityFormat, Timeout Multiplier.PI shall be equal to the smaller of Timeout_Multiplier or 4, whichever is lower.

**TST32** The Producer Timeout Multiplier Margin Test shall confirm that if a consumer never returns a Time Coordination message, the producer faults the connection within Timeout Multiplier.PI+2 Ping\_Intervals

#### **Required Initial Conditions:**

1. This test supports both the base format and the Extended Format. It will use the Timeout\_Multiplier.PI enhanced definition.
2. The DUT will be configured to support 15 consumers.
3. The Timeout\_Multiplier in the connection request will be varied from 1 to 4 for each base format consumer or 1 to 6 for Extended Format consumers.

#### **Test Procedure:**

1. A connection will be established using the base or Extended Format depending on which format is being tested.
2. The test will not send Time Coordination responses to Ping Requests
3. As the consumer connection is faulted, the test will verify that the producer detected the fault at Timeout\_Multiplier.PI+ 2 Ping\_Intervals. This will be detected by seeing that ping requests for that consumer # are no longer requested, or data is no longer produced.
4. Steps 1-3 will be repeated for all the Timeout\_Multiplier values based on format type.



### **F-3.8 Single\_Cast Consumer Negative Tests**

#### **F-3.8.1 TST33 – Base Format Single-Cast Consumer Test; >= 3 Bytes, Negative**

<b>Requirement Number</b>	<b>Requirement</b>
FRS7	The PID or CID is incorporated within the Safety CRC calculation in both the producer and the consumer, thus, if a message is mistakenly received, it shall be detected when the CRC is checked.
FRS9	The following Sequence of checks shall be used to check messages: Ping count check Evaluate Time stamp section CRC Evaluated Time Stamps Evaluate Actual Data CRC Evaluate Complement Data CRC Perform Cross-Check
FRS13	When crosschecked safety data are found to be different, the data is treated as faulty. The base format consumer shall terminate the connection and the Extended Format consumer will increment the Consumer_Fault_Count and drop the packet if the count is less than the Max_Fault_Number, or terminate the connection if is equal to or greater than the Max_Fault_Number.
FRS17	The Single-cast SafetyValidatorServer shall respond to a ping request and produces an Time Coordination message with a Time_Value to the Single-cast SafetyValidatorClient that is used by the safety data producer.
FRS48	The time coordination section shall be sent from the consumer to the producer in response to a change in the ping count.
FRS71	When the consumer sees a change in the ping count, the consumer shall prepare a Time Coordination response message that contains the consumer's value of its clock count
FRS156	The Consumer shall obtain the Producer Identifier from the SafetyOpen and use the value as an initial seed in runtime checking the safety CRCs.
FRS190	The Reserved bits shall be ignored by the consumer except for CRC checking and Actual/Complement checking of the TBD_Bit/N TBD_Bit pair.
FRS191	The N_TBD_Bit complement bit shall always be the complement of the TBD_Bit

TST33 The Base format Single-cast Consumer test shall test all the basic features of the safety protocol for messages with >3 bytes of data by injecting error in each component listed in the above requirement.

#### **Required Initial Conditions:**

1. Run SPTE as a target

#### **Test Procedure:**

1. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
2. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
3. The connection will run for a set period of time.
4. The producer will send an incorrect Time Stamp CRC.
5. The consumer will detect the error and the connection will be terminated.



6. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
7. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
8. The producer will send an out of sequence Time Stamp.
9. The consumer will detect the error and the connection will be terminated.
10. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
11. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
12. The producer will send an incorrect Data CRC.
13. The consumer will detect the error and the connection will be terminated.
14. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
15. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
16. The producer will send an incorrect Complemented Data CRC.
17. The consumer will detect the error and the connection will be terminated.
18. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
19. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
20. The producer will send a Data and Complemented Data mismatch.
21. The consumer will detect the error and the connection will be terminated.
22. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
23. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
24. The producer will send a TBD\_Bit and N\_TBD\_Bit mismatch.
25. The consumer will detect the error and the connection will be terminated
26. Re-establish the connection with the DUT
27. Valid Safety data will be sent, but with an invalid Producer ID being used to seed the safety CRC
28. The consumer will detect the error and the connection will be terminated.
29. Set up a safety I/O connection of a type supported by the DUT.
30. Invalidate the N\_Run\_Idle bit in the mode byte
31. The consumer will detect the error and the connection will be terminated.

**F-3.8.2 TST123 – Extended Format Single-Cast Consumer Test; >= 3 Bytes, Negative**

Requirement Number	Requirement
FRS7	The PID or CID is incorporated within the Safety CRC calculation in both the producer and the consumer, thus, if a message is mistakenly received, it shall be detected when the



Requirement Number	Requirement
	CRC is checked.
FRS9	The following Sequence of checks shall be used to check messages: Ping count check Evaluate Time stamp section CRC Evaluated Time Stamps Evaluate Actual Data CRC Evaluate Complement Data CRC Perform Cross-Check
FRS13	When crosschecked safety data are found to be different, the data is treated as faulty. The base format consumer shall terminate the connection and the Extended Format consumer will increment the Consumer_Fault_Count and drop the packet if the count is less than the Max_Fault_Number, or terminate the connection if is equal to or greater than the Max_Fault_Number.
FRS17	The Single-cast SafetyValidatorServer shall respond to a ping request and produces an Time Coordination message with a Time_Value to the Single-cast SafetyValidatorClient that is used by the safety data producer.
FRS48	The time coordination section shall be sent from the consumer to the producer in response to a change in the ping count.
FRS71	When the consumer sees a change in the ping count, the consumer shall prepare a Time Coordination response message that contains the consumer's value of its clock count
FRS156	The Consumer shall obtain the Producer Identifier from the SafetyOpen and use the value as an initial seed in runtime checking the safety CRCs.
FRS190	The Reserved bits shall be ignored by the consumer except for CRC checking and Actual/Complement checking of the TBD_Bit/N TBD_Bit pair.
FRS191	The N_TBD_Bit complement bit shall always be the complement of the TBD_Bit

TST123 The Extended Format Single-cast Consumer test shall test all the basic features of the Extended Format for messages with >3 bytes of data by injecting error in each component listed in the above requirement.

**Required Initial Conditions:**

1. Run SPTE as a target

**Test Procedure:**

1. Run TST29 for a Extended Format Single-cast connections with a size >3 bytes
2. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
3. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
4. The producer will send a Data and Complemented Data mismatch for Max\_Fault\_Count – 1 times.
5. Verify that the packet is dropped and connection continued.
6. Send a data packet with mismatched data one more time
7. Verify that the connection is terminated.
8. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.



9. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
10. The producer will send a TBD\_Bit and N\_TBD\_Bit mismatch for Max\_Fault\_Count – 1 times.
11. Verify that the packet is dropped and connection continued.
12. Send a data packet with send a TBD\_Bit and N\_TBD\_Bit mismatch one more time
13. Verify that the connection is terminated
14. Re-establish the connection with the DUT
15. Valid Safety data will be sent, but with an invalid Producer ID being used to seed the safety CRC for Max\_Fault\_Count – 1 times.
16. Verify that the packet is dropped and connection continued.
17. Send a data packet with invalid Producer ID one more time
18. Verify that the connection is terminated
19. Set up a safety I/O connection of a type supported by the DUT.
20. Invalidate the N\_Run\_Idle bit in the mode byte for Max\_Fault\_Count – 1 times.
21. Verify that the packet is dropped and connection continued.
22. Send a data packet with invalid N\_Run\_Idle bit one more time
23. Verify that the connection is terminated
24. Re-establish the connection with the DUT
25. Valid Safety data will be sent, but with an invalid Rollover Count being used to seed the safety CRC for Max\_Fault\_Count – 1 times.
26. Verify that the packet is dropped and connection continued.
27. Send a data packet with invalid Rollover Count one more time
28. Verify that the connection is terminated

**F-3.8.3 TST34 – Base format Single-Cast Consumer; <= 2 Bytes, Negative**

Requirement Number	Requirement
FRS7	The PID or CID is incorporated within the Safety CRC calculation in both the producer and the consumer, thus, if a message is mistakenly received, it shall be detected when the CRC is checked.
FRS9	The following Sequence of checks shall be used to check messages: Ping count check Evaluate Time stamp section CRC Evaluated Time Stamps Evaluate Actual Data CRC Evaluate Complement Data CRC Perform Cross-Check (when appropriate)
FRS17	The Single-cast SafetyValidatorServer shall respond to a ping request and produces an Time Coordination message with a Time_Value to the Single-cast SafetyValidatorClient that is used by the safety data producer.
FRS48	The time coordination section shall be sent from the consumer to the producer in response to a change in the ping count.
FRS71	When the consumer sees a change in the ping count, the consumer shall prepare a Time Coordination response message that contains the consumer's value of its clock count



<b>Requirement Number</b>	<b>Requirement</b>
FRS156	The Consumer shall obtain the Producer Identifier from the SafetyOpen and use the value as an initial seed in runtime checking the safety CRCs.
FRS190	The Reserved bits shall be ignored by the consumer except for CRC checking and Actual/Complement checking of the TBD_Bit/N_TBD_Bit pair.
FRS191	The N_TBD_Bit complement bit shall always be the complement of the TBD_Bit

TST34 The Base format Single-cast Consumer test shall test all the basic features of the safety protocol for message < 3 bytes in size by injecting error in each component listed in the requirements

**Required Initial Conditions:**

1. Run SPTE as a target

**Test Procedure:**

1. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 1 or 2 bytes.
2. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
3. The connection will run for a set period of time.
4. The producer will send an incorrect Time Stamp CRC.
5. The consumer will detect the error and the connection will be terminated.
6. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 2 bytes or smaller.
7. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
8. The producer will send an out of sequence Time Stamp.
9. The consumer will detect the error and the connection will be terminated.
10. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 2 bytes or smaller.
11. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
12. The producer will send an incorrect Data CRC.
13. The consumer will detect the error and the connection will be terminated.
14. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 2 bytes or smaller.
15. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
16. The producer will send a TBD\_Bit and N\_TBD\_Bit mismatch.
17. The consumer will detect the error and the connection will be terminated
18. Re-establish the connection with the DUT
19. Valid Safety data will be sent, but with an invalid Producer ID being used to seed the safety CRC



20. The consumer will detect the error and the connection will be terminated.
21. Set up a safety I/O connection of a type supported by the DUT.
22. Invalidate the N\_Run\_Idle bit in the mode byte
23. The consumer will detect the error and the connection will be terminated.

**F-3.8.4 TST124 – Extended Format Single-Cast Consumer; <= 2 Bytes, Negative**

Requirement Number	Requirement
FRS7	The PID or CID is incorporated within the Safety CRC calculation in both the producer and the consumer, thus, if a message is mistakenly received, it shall be detected when the CRC is checked.
FRS9	The following Sequence of checks shall be used to check messages: Ping count check Evaluate Time stamp section CRC Evaluated Time Stamps Evaluate Actual Data CRC Evaluate Complement Data CRC Perform Cross-Check (when appropriate)
FRS17	The Single-cast SafetyValidatorServer shall respond to a ping request and produces an Time Coordination message with a Time Value to the Single-cast SafetyValidatorClient that is used by the safety data producer.
FRS48	The time coordination section shall be sent from the consumer to the producer in response to a change in the ping count.
FRS71	When the consumer sees a change in the ping count, the consumer shall prepare a Time Coordination response message that contains the consumer's value of its clock count
FRS156	The Consumer shall obtain the Producer Identifier from the SafetyOpen and use the value as an initial seed in runtime checking the safety CRCs.
FRS190	The Reserved bits shall be ignored by the consumer except for CRC checking and Actual/Complement checking of the TBD_Bit/N_TBD_Bit pair.
FRS191	The N_TBD_Bit complement bit shall always be the complement of the TBD_Bit

TST124 The Extended Format Single-cast Consumer test shall test all the basic features of the safety protocol for message < 3 bytes in size by injecting error in each component listed in the requirements

**Required Initial Conditions:**

1. Run SPTE as a target

**Test Procedure:**

1. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 1 or 2 bytes.
2. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
3. The connection will run for a set period of time.
4. The producer will send an out of sequence Time Stamp for Max\_Fault\_Count – 1 times.
5. Verify that the packet is dropped and connection continued.
6. Send a data packet with an out of sequence Time Stamp one more time



7. Verify that the connection is terminated.
8. The consumer will detect the error and the connection will be terminated.
9. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 2 bytes or smaller.
10. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
11. The producer will send an incorrect Data CRC for Max\_Fault\_Count – 1 times.
12. Verify that the packet is dropped and connection continued.
13. Send a data packet with an incorrect Data CRC more time
14. Verify that the connection is terminated.
15. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 2 bytes or smaller.
16. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
17. The producer will send a TBD\_Bit and N\_TBD\_Bit mismatch for Max\_Fault\_Count – 1 times.
18. Verify that the packet is dropped and connection continued.
19. Send a data packet with a TBD\_Bit and N\_TBD\_Bit mismatch one more time
20. Verify that the connection is terminated.
21. Re-establish the connection with the DUT
22. Valid Safety data will be sent, but with an invalid Producer ID being used to seed the safety CRC for Max\_Fault\_Count – 1 times.
23. Verify that the packet is dropped and connection continued.
24. Send a data packet with an invalid Producer ID one more time
25. Verify that the connection is terminated.
26. Set up a safety I/O connection of a type supported by the DUT.
27. Invalidate the N\_Run\_Idle bit in the mode byte for Max\_Fault\_Count – 1 times
28. Verify that the packet is dropped and connection continued.
29. Send a data packet with an invalid N\_Run\_Idle bit in the mode byte one more time
30. Verify that the connection is terminated.
31. Set up a safety I/O connection of a type supported by the DUT.
32. Send a data packet with an invalid rollover count in the data CRC for Max\_Fault\_Count – 1 times
33. Verify that the packet is dropped and connection continued.
34. Send a data packet with an invalid rollover count in the data CRC one more time
35. Verify that the connection is terminated.

**F-3.8.5 TST36 - Single-Cast Consumer Communication Failure**

Requirement Number	Requirement
FRS11	All consumers shall monitor the periodic transmission of data and go to a safety state if the periodic transmissions cease.



<b>Requirement Number</b>	<b>Requirement</b>
FRS12	<p>When the Safety Layer detects an error requiring connection termination the termination shall be implemented using the following sequence.</p> <p>The safety layer detects an error requiring termination</p> <p>The safety layer notifies the application program by setting the connection status to indicate a safety communications fault</p> <p>The safety application shall transition data and I/O (e.g. set outputs to the safety state) associated with the connection to a safety state</p> <p>The safety layer shall notify the underlying communications system of an error and request the termination of the connection by setting its status to indicate a safety communications error</p> <p>The safety layer shall not transition from its safety state until the fault is cleared and a connection restart sequence is initiated</p>
FRS78	Link-triggered consumers shall maintain an activity timer, which is reset after receiving a valid expected message
FRS79	If a valid message has not been received before the activity timer expires, the link-triggered consumer shall terminate the connection (See 2-1.3.2) and go to the defined safety state.
FRS81	If the age of the data is greater than the network time expectation the consumer shall terminate the connection (See 2-1.3.2) and go to the defined safety state.
FRS287	For all consumed safety connections, time stamp checking shall be enabled if the mode of the data indicates Run.

TST36 The Single-cast Consumer Communication Failure test shall confirm that consumer DUTs perform safe state behavior on communication failures.

**Required Initial Conditions:**

1. This test supports both the base format and the Extended Format. It will use the Timeout\_Multiplier.PI enhanced definition.
2. Run SPTE as a target

**Test Procedure:**

1. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 1 or 2 bytes.
2. The establishment of a safety I/O connection is a positive test case for the date age and time stamp checking.
3. The producer will send data with the Time Stamp value set to 0 and the Run\_Idle bit set to IDLE
4. The data will be sent at such a rate that the age detection limit will be exceeded.
5. Confirm that the connection remains established.
6. The producer will send data with the Time Stamp value set to 0 and the Run\_Idle bit set to RUN
7. The data will be sent at such a rate that the age detection limit will be exceeded.
8. The consumer will detect the error and the connection will be terminated



### **F-3.9 Multi-Cast Connections**

This section will define tests specifically related to Multi-Cast Connections. It defines producer and consumer tests separately since the test setups will be unique for each.

#### **F-3.9.1 TST37 – Base Format Multi-Cast Consumer Negative**

<b>Requirement Number</b>	<b>Requirement</b>
FRS9	The following Sequence of checks shall be used to check messages: Ping count check Evaluate Time stamp section CRC Evaluated Time Stamps Evaluate Actual Data CRC Evaluate Complement Data CRC Perform Cross Check
FRS13	When crosschecked safety data are found to be different, the data is treated as faulty. The base format consumer shall terminate the connection (See Section 2-1.3.2) and the Extended Format consumer will increment the Consumer_Fault_Count and drop the packet if the count is less than the Max_Fault_Number, or terminate the connection if is equal to or greater than the Max_Fault_Number.
FRS190	The Reserved bits shall be ignored by the consumer except for CRC checking and Actual/Complement checking of the TBD_Bit/N_TBD_Bit pair
FRS191	The N_TBD_Bit complement bit shall always be the complement of the TBD_Bit

TST37 The Base Format Multi-cast Consumer test shall test all the basic features of the safety protocol by injecting errors in each component listed in the above requirement

#### **Required Initial Conditions:**

1. Run SPTE as a target

#### **Test Procedure:**

1. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
2. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
3. The connection will run for a set period of time.
4. The producer will send an incorrect Time Stamp CRC.
5. The consumer will detect the error and the connection will be terminated.
6. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
7. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
8. The producer will send an out of sequence Time Stamp.
9. The consumer will detect the error and the connection will be terminated.
10. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.



11. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
12. The producer will send an incorrect Data CRC.
13. The consumer will detect the error and the connection will be terminated.
14. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
15. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
16. The producer will send an incorrect Complemented Data CRC.
17. The consumer will detect the error and the connection will be terminated.
18. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
19. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
20. The producer will send a Data and Complemented Data mismatch.
21. The consumer will detect the error and the connection will be terminated.
22. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
23. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
24. The producer will send a TBD\_Bit and N\_TBD\_Bit mismatch.
25. The consumer will detect the error and the connection will be terminated.

### **F-3.9.2 TST117 – Extended Format Multi-Cast Consumer Negative**

<b>Requirement Number</b>	<b>Requirement</b>
FRS9	The following Sequence of checks shall be used to check messages: Ping count check Evaluate Time stamp section CRC Evaluated Time Stamps Evaluate Actual Data CRC Evaluate Complement Data CRC Perform Cross Check
FRS13	When crosschecked safety data are found to be different, the data is treated as faulty. The base format consumer shall terminate the connection and the Extended Format consumer will increment the Consumer_Fault_Count and drop the packet if the count is less than the Max_Fault_Number, or terminate the connection if is equal to or greater than the Max_Fault_Number.
FRS190	The Reserved bits shall be ignored by the consumer except for CRC checking and Actual/Complement checking of the TBD_Bit/N_TBD_Bit pair
FRS191	The N_TBD_Bit complement bit shall always be the complement of the TBD_Bit
FRS378	The active seeding of the CRC with the rollover count in Extended Format Multi-cast messages shall begin immediately on first production.

TST117 The Extended Format Multi-cast Consumer test shall test all the basic features of the safety protocol by injecting errors in each component listed in the above requirement



**Required Initial Conditions:**

1. Run SPTE as a target

**Test Procedure:**

1. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
2. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
3. The connection will run for a set period of time.
4. Read the Producer/Consumer Fault Count Attribute in Safety Validator and confirm it is zero.
5. The producer will send an incorrect Complement Data CRC.
6. The consumer will detect the error and the message will be dropped.
7. Read the Producer/Consumer Fault Count Attribute in Safety Validator instance and confirm it is non-zero but less than Max\_Fault\_Count
8. Repeat the production 4 more times
9. The consumer will detect the error and the message will be dropped for production 2, 3, & 4 and close the connection after the 5<sup>th</sup> production
10. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
11. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
12. The producer will send an out of sequence Time Stamp.
13. The consumer will detect the error and the message will be dropped.
14. Read the Producer/Consumer Fault Count Attribute in Safety Validator instance and confirm it is non-zero but less than Max\_Fault\_Count
15. Repeat the production 4 more times
16. The consumer will detect the error and the message will be dropped for production 2, 3, & 4 and close the connection after the 5<sup>th</sup> production
17. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
18. The producer will send a message using an invalid Rollover count.
19. The consumer will detect the error and the message will be dropped.
20. Read the Producer/Consumer Fault Count Attribute in Safety Validator instance and confirm it is non-zero but less than Max\_Fault\_Count
21. Repeat the production 4 more times
22. The consumer will detect the error and the message will be dropped for production 2, 3, & 4 and close the connection after the 5<sup>th</sup> production
- 23.
24. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.



25. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
26. The producer will send an incorrect Data CRC.
27. The consumer will detect the error and the message will be dropped.
28. Read the Producer/Consumer Fault Count Attribute in Safety Validator instance and confirm it is non-zero but less than Max\_Fault\_Count
29. Repeat the production 4 more times
30. The consumer will detect the error and the message will be dropped for production 2, 3, & 4 and close the connection after the 5<sup>th</sup> production
31. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
32. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
33. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
34. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
35. The producer will send a Data and Complemented Data mismatch.
36. The consumer will detect the error and the message will be dropped.
37. Read the Producer/Consumer Fault Count Attribute in Safety Validator instance and confirm it is non-zero but less than Max\_Fault\_Count
38. Repeat the production 4 more times
39. The consumer will detect the error and the message will be dropped for production 2, 3, & 4 and close the connection after the 5<sup>th</sup> production
40. Set up a safety I/O connection of a type supported by the DUT. The connection size will be 3 bytes or larger.
41. The establishment of a safety I/O connection is a positive test case when the DUT is a consumer.
42. The producer will send a TBD\_Bit and N\_TBD\_Bit mismatch.
43. The consumer will detect the error and the message will be dropped.
44. Read the Producer/Consumer Fault Count Attribute in Safety Validator instance and confirm it is non-zero but less than Max\_Fault\_Count
45. Repeat the production 4 more times
46. The consumer will detect the error and the message will be dropped for production 2, 3, & 4 and close the connection after the 5<sup>th</sup> production

**F-3.9.2.1 TST38 – Base Format Multi-cast Consumer PID**

Requirement Number	Requirement
FRS7	The PID or CID is incorporated within the Safety CRC calculation in both the producer and the consumer, thus, if a message is mistakenly received, it shall be detected when the CRC is checked.
FRS155	The Producer Identifier (PID) shall be used as an initial seed value in the CRCs in the runtime protocol



<b>Requirement Number</b>	<b>Requirement</b>
FRS156	The Consumer shall obtain the Producer Identifier from the SafetyOpen and use the value as an initial seed in runtime checking the safety CRCs.

**TST38** The Multi-cast Consumer PID Test shall confirm that a multi-cast consumer DUT will include the producer ID in the CRC calculation and detect if data is received from an unintended source.

**Required Initial Conditions:**

1. Run SPTE as a target

**Test Procedure:**

1. The DUT will be configured with a representative configuration
2. The DUT will be triggered to send SafetyOpen and an initial Producer ID will be returned
3. Valid safety data will be sent and the test will confirm that the data is accepted
4. Valid Safety data will be sent, but with an invalid Producer ID being used to seed the safety CRC
5. The test will confirm that the DUT detects the invalid CRC, faults the connection, notifies the application, and safety states the consumed data.

**F-3.9.2.2 TST125 – Extended Format Multi-cast Consumer PID/Rollover**

<b>Requirement Number</b>	<b>Requirement</b>
FRS7	The PID or CID is incorporated within the Safety CRC calculation in both the producer and the consumer, thus, if a message is mistakenly received, it shall be detected when the CRC is checked
FRS155	The Producer Identifier (PID) shall be used as an initial seed value in the CRCs in the runtime protocol
FRS156	The Consumer shall obtain the Producer Identifier from the SafetyOpen and use the value as an initial seed in runtime checking the safety CRCs.
FRS378	The active seeding of the CRC with the rollover count in Extended Format Multi-cast messages shall begin immediately on first production.

**TST125** The Extended Format Multi-cast Consumer PID/Rollover Test shall confirm that a multi-cast consumer DUT will include the producer ID and proper rollover count in the CRC calculation and detect if data is received from an unintended source.

**Required Initial Conditions:**

1. Run SPTE as a target

**Test Procedure:**

1. The DUT will be configured with a representative configuration
2. The DUT will be triggered to send SafetyOpen and an initial Producer ID, Initial Time Stamp, and Initial Rollover count will be returned
3. Valid safety data will be sent and the test will confirm that the data is accepted



4. Valid Safety data will be sent, but with an invalid Producer ID being used to seed the safety CRC for Max\_Fault\_Count – 1 times.
5. Verify that the packet is dropped and connection continued.
6. Send a data packet with an invalid Producer ID one more time
7. Verify that the connection is terminated.
8. Valid Safety data will be sent, but with an invalid Rollover Count being used to seed the safety CRC for Max\_Fault\_Count – 1 times.
9. Verify that the packet is dropped and connection continued.
10. Send a data packet with an invalid Rollover Count one more time
11. Verify that the connection is terminated
12. The test will confirm that the DUT detects the invalid CRC, faults the connection, notifies the application, and safety states the consumed data.

**F-3.9.2.3 TST39 - Multi-cast Consumer Communication Failure Response**

Requirement Number	Requirement
FRS11	All consumers shall monitor the periodic transmission of data and go to a safety state if the periodic transmissions cease
FRS12	When the Safety Layer detects an error requiring connection termination the termination may be implemented using the following sequence <ul style="list-style-type: none"> <li>• The safety layer detects an error requiring termination</li> <li>• The safety layer notifies the application program by setting the connection status to indicate a safety communications error</li> <li>• The safety application shall transition data and I/O associated with the connection to a safety state</li> <li>• The safety layer shall notify the underlying communications system of an error and request the termination of the connection by setting its status to indicate a safety communications error</li> <li>• The safety layer shall not transition from its safety state until the fault is cleared and a connection restart sequence is initiated.</li> </ul>
FRS78	Link-triggered consumers shall maintain an activity timer, which is reset after receiving a valid expected message
FRS79	If a valid message has not been received before the activity timer expires, the link-triggered consumer shall terminate the connection and go to the defined safety state.
FRS81	If the age of the data is greater than the network time expectation the consumer shall terminate the connection (See 2-1.3.2) and go to the defined safety state.
FRS287	For all consumed safety connections, time stamp checking shall be enabled if the mode of the data indicates Run.

TST39 The Multi-cast Consumer Communication Failure test shall confirm that consumer DUTs perform safe state behavior on communication failures.

**Required Initial Conditions:**

1. This will function for both Base and Extended Format connections
2. Run SPTE as a target

**Test Procedure:**

1. Send a Time Correction message at the appropriate time



2. Generate a safety message with a Time Stamp that exceeds the age limit based on configured timeout parameters
3. Confirm that the DUT faults the connection and stops responding to Ping Requests and begins requesting a new connection
4. Allow the DUT to re-establish the connection
5. Break communication with the device
6. Allow the DUT to re-establish the connection
7. Don't send a Time Correction Message
8. Confirm that the DUT faults the connection and stops responding to Ping Requests and begins requesting a new connection
9. Allow the DUT to re-establish the connection
10. Produce data with Run\_Idle = Run, but at an EPI rate that violates the age limit
11. Confirm the DUT detects the age violation and faults the connection

**F-3.9.2.4 TST40 – Base Format Multi-cast Consumer Time Correction Negative**

Requirement Number	Requirement
FRS62	Parity Even in the Time Correction message shall be the even parity of bits 0 through 6 of the Message
FRS92	For DeviceNet Multi-cast connections, a separate message shall be used to communicate time correction information from the produce to each multi-cast consumer.
FRS107	The consumer shall remain in a safety state until the initial ping sequence is successfully completed
FRS206	The Multi_Cast_Active_Idle, and the Consumer_Time_Correction_Value shall be applied by the consumer indicated by the Consumer_#.
FRS209	The Multi_Cast_Active_Idle bit value of 0 shall indicate Idle, and the Multi_Cast_Active_Idle bit value of 1 shall indicate Active.
FRS211	If the consumer sees the Multi-Cast_Active_Idle bit transition from Active to Idle, it shall close the connection if the base format is used. If the Extended Format is used and the Max_Fault_Count has not been reached, the packet will be dropped, otherwise the connection will be closed.
FRS216	The Reserved bits shall be ignored by the consumer except for CRC checking and MC_Byte_2 checking
FRS305	If the Time_Correction_Ping_Interval_Count is greater than the Timeout_Multiplier.PI + 1, the connection shall be faulted. This requirement is confirmed indirectly
FRS374	For the Base Format, Timeout_Multiplier.PI shall be equal to the Timeout_Multiplier. For the HiAvailabilityFormat, Timeout_Multiplier.PI shall be equal to the smaller of Timeout_Multiplier or 4, whichever is lower.

TST40 The Base Format Multi-cast Consumer Time Correction Processing Test shall confirm that the Multi-cast Consumer DUT is processing the Time Correction Message properly.

**Required Initial Conditions:**

1. Run SPTE as an originating target

**Test Procedure:**

1. Issue a ping request



2. Process the Time Coordination response and construct a EF or Base format Time Correction message
3. Send a valid Time correction message with Multi\_Cast\_Active\_Idle = Active = 1
4. Confirm that the multi-cast consumer DUT establishes the safety connection and applies the data
5. Send additional time correction message with Consumer # other than that of the DUT with Multi\_Cast\_Active\_Idle = IDLE
6. Confirm the multi-cast consumer DUT ignores these messages
7. Start another ping cycle
8. Inject an invalid Parity-bit in the Time Correction message
9. Confirm that the multi-cast consumer DUT faults the connection.
10. Allow the multi-cast consumer DUT to re-establish the connection
11. During this ping cycle, send a Time Correction message with **non-zero** reserved bits (with valid CRC and MC\_Byte\_2)
12. Confirm the multi-cast consumer DUT accepts this time correction as valid
13. Complete a valid ping cycle including a valid Time Correction with Multi\_Cast\_Active\_Idle = Active (1)
14. During the next ping cycle send a valid Time correction message with Multi\_Cast\_Active\_Idle = Idle (0)
15. Confirm the multi-cast consumer DUT faults the connection
16. Allow the multi-cast consumer DUT to re-establish the connection
17. Complete a valid ping cycle to fully establish the connection
18. Stop sending Time Correction messages to the DUT but continue sending data and ping requests normally
19. Confirm the multi-cast consumer DUT faults the connection after Timeout\_Multiplier.PI +1 Ping Intervals

**F-3.9.2.5 TST126 – Extended Format Multi-cast Consumer Time Correction Negative**

Requirement Number	Requirement
FRS62	Parity Even in the Time Correction message shall be the even parity of bits 0 through 6 of the Message
FRS92	For DeviceNet Multi cast connections, a separate message shall be used to communicate time correction information from the produce to each multi-cast consumer.
FRS107	The consumer shall remain in a safety state until the initial ping sequence is successfully completed.
FRS206	The Multi_Cast_Active_Idle, and the Consumer_Time_Correction_Value shall be applied by the consumer indicated by the Consumer_#.
FRS209	The Multi_Cast_Active_Idle bit value of 0 shall indicate Idle, and the Multi_Cast_Active_Idle bit value of 1 shall indicate Active.
FRS211	If the consumer sees the Multi-Cast_Active_Idle bit transition from Active to Idle, it shall close the connection if the base format is used. If the Extended Format is used and the Max_Fault_Count has not been reached, the packet will be dropped, otherwise the connection will be closed
FRS216	The Reserved bits shall be ignored by the consumer except for CRC checking and MC_Byte_2 checking



Requirement Number	Requirement
FRS305	If the Time_Correction_Ping_Interval_Count is greater than the Timeout_Multiplier.PI + 1, the connection shall be faulted This requirement is confirmed indirectly
FRS374	For the Base Format, Timeout_Multiplier.PI shall be equal to the Timeout_Multiplier. For the HiAvailabilityFormat, Timeout_Multiplier.PI shall be equal to the smaller of Timeout_Multiplier or 4, whichever is lower.

TST126 The Extended Format Multi-cast Consumer Time Correction Processing Test shall confirm that the Multi-cast Consumer DUT is processing the Time Correction Message properly.

**Required Initial Conditions:**

1. Run SPTE as an originating target

**Test Procedure:**

1. Issue a ping request
2. Process the Time Coordination response and construct a Extended Format Time Correction message
3. Send a valid Time correction message with Multi\_Cast\_Active\_Idle = Active = 1
4. Confirm that the multi-cast consumer DUT establishes the safety connection and applies the data
5. Send additional time correction message with Consumer # other than that of the DUT with Multi\_Cast\_Active\_Idle = IDLE
6. Confirm the multi-cast consumer DUT ignores these messages
7. Start another ping cycle
8. Send Time Correction packets with an invalid Parity-bit for (Max\_Fault\_Count) or (Timeout\_Multiplier.PI+1 Ping Intervals); whichever is less.
9. Verify that the DUT closes the connection after the last transmission but not the prior ones.
10. Allow the multi-cast consumer DUT to re-establish the connection
11. During this ping cycle, send a Time Correction message with **non-zero** reserved bits (with valid CRC)
12. Confirm the multi-cast consumer DUT accepts this time correction as valid
13. Complete a valid ping cycle including a valid Time Correction with Multi\_Cast\_Active\_Idle = Active (1)
14. During the next ping cycle send a valid Time correction message with Multi\_Cast\_Active\_Idle = Idle (0) for (Max\_Fault\_Count) or (Timeout\_Multiplier.PI+1 Ping Intervals); whichever is less.
15. Confirm the multi-cast consumer DUT faults the connection after the last transmission but not the prior ones.
16. Allow the multi-cast consumer DUT to re-establish the connection
17. Complete a valid ping cycle to fully establish the connection



18. Stop sending Time Correction messages to the DUT but continue sending data and ping requests normal
19. Confirm the multi-cast consumer DUT faults the connection after Timeout\_Multiplier.PI +1 Ping Intervals

**F-3.9.2.6 TST41 - Multi-cast Producer Time Correction Generation Negative**

Requirement Number	Requirement
FRS65	The Consumer_# in the Time Correction message shall indicates the number of the consumer that this message is directed to.
FRS179	The Run_Idle bit value of 0 shall indicate Idle, and the Run_Idle bit value of 1 shall indicate Run
FRS205	For multi-cast produced data the Multi_Cast_Active_Idle, and Consumer Time Correction Value shall be directed to the consumer indicated by the Consumer_# field of the message.
FRS209	The Multi_Cast_Active Idle bit value of 0 shall indicate Idle, and the Multi_Cast_Active_Idle bit value of 1 shall indicate Active.
FRS210	Once the Multi_Cast_Active Idle bit has been set to Active after 1st data production, this bit shall not be set back to idle until the safety connection is re-initialized
FRS358	Multi_Cast_Active_Idle in the Time Correction message shall indicate Idle if transmitted before this consumer has responded with a valid time coordination message

TST41 The Multi-cast Producer Time Correction Generation Test shall generate a number of conditions to show the producer is generating the Time Correction message correctly.

**Initial Conditions:**

1. This test supports both the base format and the Extended Format. It will use the Timeout\_Multiplier.PI enhanced definition.
2. Run the SPTE as an originator of either the EF or Base format connections

**Procedure:**

1. Note the assigned Producer Id
2. Confirm a Ping request is generated at the assigned consumer #
3. Send the appropriate Time Coordination responses at the appropriate time
4. Run TST18 - Producer Time Correction CRC
5. Confirm that the Time Correction messages has Multi\_Cast\_Run\_Idle is "Run (=1)"
6. Run TST19 - Producer Time Correction Mcast Byte & Mcast Byte 2 if the Base format is used or TST111 if the Extended Format is used.
7. Confirm the Consumer # matches the number assigned to the tester
8. On the next ping response, generate a EF or Base format Time Coordination message with a bad CRC
9. Confirm the Time Correction messages are no longer sent
10. Re-establish the connection
11. After one good ping cycle, send a Time Coordination response with a bad Ack\_Byte
12. Confirm the Time Correction messages are no longer sent
13. Re-establish the connection



14. After one good ping cycle, stop sending Time Coordination responses
15. Confirm the Time Correction messages are no longer sent
16. if some Time Correction message get through before detection, confirm the  
Multi\_Cast\_Active\_Idle = Active = 1

### **F-3.10 Device Configuration**

#### **F-3.10.1 Target Configuration**

The DUT in this section will be a target (receiver) of connection requests. The requirements called out here will determine if the server is properly handling the Type 1 Safety\_Forward\_Open message over a Class 3 connection.

There are some tests that must be done regardless of the types of connections supported. However, these tests will need to be executed as part of the connection tests for a supported type. Therefore the tests defined here should be considered generic and must be adapted to work as part of the test suite for the supported connection types.

##### **F-3.10.1.1 Configuration Ownership**

These tests will verify all Server behavior associated with connection ownership.

##### **F-3.10.1.2 TST42 - Configuration UNID**

<b>Requirement Number</b>	<b>Requirement</b>
FRS165	Any attempts to reconfigure a device via the SafetyOpen shall be rejected if the tool only configuration mode is selected.
SRS70	The Configuration OUNID attribute shall have the following special meanings assigned; 0 = No owner, accept any all 0xFF = Software Tool is the assigned owner
SRS205	When the Configuration UNID attribute is set to "No owner, accept any", CIP Safety Devices shall capture the first configuration source (Type 1 and/or Configure_Request) as the owner

TST42 The Configuration UNID test shall confirm that the server DUT will behave properly with the various OUNID values.

#### **Initial Conditions:**

1. This test supports both the base format and the Extended Format. The Extended Format will use the EF Safety Segment type
2. Run the SPTE as an originator of either the EF or Base format connections

#### **The test should do the following:**

1. Reset the DUT to an out-of-box condition (the device should initialize the Config. UNID to 0, accept any)
2. Open a class 3 connection
3. Set the TUNID to get the device into Configure mode
4. If the DUT support configuration via Type 1 connection requests, send a Type 1 w/valid configuration using any OUNID



5. Confirm the DUT accepts the configuration and connects
6. Reset the DUT back to out-of-box condition and set the TUNID
7. Send a Configure\_Request with a OUNID = 0xFF.
8. Complete the configuration with a valid configuration.
9. Open a Class 3 connection
10. Send the DUT a completely valid Type 1 SafetyOpen to an input or output assembly with any convenient OUNID
11. confirm that the target responds with the error code **0x01, 0x105** "Ownership conflict or OUNID mismatch "
12. Send the DUT a completely valid Type 1 SafetyOpen with OUNID=all 0xFFs
13. confirm that the target responds with the error code **0x01, 03** "Configuration not allowed (configuration locked)"

#### **F-3.10.1.3 TST43 - Input Type 1 Connection Establishment**

<b>Requirement Number</b>	<b>Requirement</b>
FRS100	It shall be permissible to combine the device and the connection configuration.
FRS167	The OUNID of the originating device shall be stored in non-volatile storage with the configuration to which it is associated.
FRS164	If the TUNID to UNID comparison is successful than the SafetyOpen destination is deemed to be correct and processing shall continue.
SRS4	Safety Devices shall NV-store the OUNID for the configuration
SRS7	The originator of a Type 1 SafetyOpen that configures a previously unconfigured & unowned input device shall become the owner of the configuration.
SRS11	<p>A target input device that has an existing configuration shall (at a minimum) do the following when a type 1 Safety Open is received:</p> <p>Verify the CPCRC over the connection parameters is correct,</p> <p>Verify that the TUNID in the SafetyOpen matches the device TUNID</p> <p>Verify the Electronic Key matches the device</p> <p>If configuration data is included (type 1), verify that the OUNID in the SafetyOpen matches the stored OUNID</p> <p>Calculate the SCCRC over the received data and confirm that it obtains the same value</p> <p>Verify and validate the connection parameters,</p> <p>Validate the application path</p> <p>Confirm the safety connection requested is supported</p> <p>Safety parameters are within valid ranges</p> <p>Start the reconfiguration process (enter Config State), close and inhibit connections during the reconfiguration (respond to connection requests with an error response, "Invalid Mode/State"),</p> <p>Update the configuration and SCID in NV Memory,</p> <p>Transition the connection to the established state, and the device to the executing state.</p> <p>New connections can now be accepted</p>
SRS171	When a device receives a type 1 SafetyOpen containing configuration data and an SCID that matches its existing configuration, it shall re-apply the configuration. All existing rules checks (i.e. OUNID and TUNID checks) shall still apply.
SRS173	Targets shall respond to a Type 2 SafetyOpen with an error code 0x01, additional code 0x0110 (device not configured), when the device is not configured.



TST43 The Configuration Ownership Retention test shall verify that the server DUT performs all the basic tests and steps when processing a Type 1 safety open.

**Initial Conditions:**

1. This test supports both the base format and the Extended Format. The Extended Format will use the EF Safety Segment type
2. Run the SPTE as an originator of either the EF or Base format connections

**This test will be performed as follows:**

1. Preset the DUT to an out-of-box condition and set the TUNID
2. Send a Type 2b SafetyOpen (null SCID).
3. Confirm the DUT sends an error 0x01, 0x110 to indicate it needs to be configured.
4. Send a Type 1 SafetyOpen with valid configuration, SCID, TUNID and preset OUNID (this should establish ownership in the device)
5. Power cycle the device
6. Re-send the SafetyOpen with the same configuration, but with incorrect TUNID
7. Confirm the DUT responds with error 01 03, TUNID mismatch
8. Re-send the SafetyOpen with the same configuration, but with another OUNID
9. Confirm the DUT responds with error 01 105, CFUNID mismatch
10. Re-send the SafetyOpen with the same configuration, valid TUNID, OUNID, but with an invalid SCCRC
11. Confirm the DUT responds with error 01 x80C, SCID mismatch
12. Re-send the SafetyOpen with the same configuration, valid TUNID, OUNID, SCID, and CPCRC with a bad connection path
13. Confirm the DUT responds with error 01 117, Connection Path Error
14. Re-send the SafetyOpen with NEW configuration and SCID, but the base OUNID.
15. Confirm that the SafetyOpen is accepted and the DUT adopts the new configuration.
16. Confirm the safety I/O connection gets established

**F-3.10.1.4 TST44 - Output Type 1 Connection Establishment**

Requirement Number	Requirement
FRS100	It shall be permissible to combine the device and the connection configuration
FRS164	If the TUNID to UNID comparison is successful than the SafetyOpen destination is deemed to be correct and processing shall continue
FRS168	Outputs devices shall store the OUNID associated with safety outputs to prevent other originators from hijacking outputs inadvertently.
SRS6	Output devices shall NV-store the OUNID associated with the safety output connection.
SRS8	A Type 1 SafetyOpen that configures a previously unconfigured & unowned output device shall become the owner of both the connection and the configuration
SRS9	If a Type 1 SafetyOpen is sent where the originator OUNID is different than the OUNID that is stored with the connection the SafetyOpen shall be rejected and an error message, "O_UNID Mismatch" is returned
SRS12	A target output device that has an existing configuration shall (at a minimum) do the following when a type 1 Safety Open is received.



Requirement Number	Requirement
	<p>Verify the CPCRC over the configuration is correct,</p> <p>Verify that the TUNID in the SafetyOpen matches the device TUNID</p> <p>Verify the Electronic Key matches the device</p> <p>If configuration data is included (type 1), verify that the OUNID in the SafetyOpen matches the stored OUNID for the configuration</p> <p>If the connection path is to an output resource, verifies that the OUNID in the SafetyOpen matches the OUNID for the connection,</p> <p>Calculate the SCCRC over the received data and confirm that it obtains the same value</p> <p>Verify and validate the connection parameters,</p> <p>Validate the application path</p> <p>Confirm the safety connection requested is supported</p> <p>Safety parameters are within valid ranges</p> <p>Start the reconfiguration process by closing and inhibiting connections during the reconfiguration (respond to connection requests with an error response, "Invalid Mode/State"),</p> <p>Update the configuration and SCID in NV memory,</p> <p>Transition the connection to the established state, and the device to the executing state.</p> <p>New connections can now be accepted</p>
SRS135	The Output Connection Owner attribute shall be supported for all safety output assemblies.
SRS173	Targets shall respond to a Type 2 SafetyOpen with an error code 0x01, additional code 0x0110 (device not configured), when the device is not configured.

**TST44** The Output Type 1 Connection Establishment test shall verify that server DUT with safety outputs performs all the basic tests and steps when processing a Type 1 safety open.

**Initial Conditions:**

1. This test supports both the base format and the Extended Format. The Extended Format will use the EF Safety Segment type
2. Run the SPTE as an originator of either the EF or Base format connections

**This test will be performed as follows:**

1. Preset the DUT to an out of box condition and set the TUNID
2. Send a Type 2b SafetyOpen (null SCID).
3. Confirm the DUT sends an error 0x01, 0x110 to indicate it needs to be configured.
4. Send a Type 1 SafetyOpen with valid configuration, SCID, TUNID and preset OUNID (this should establish ownership in the device)
5. Power cycle the device
6. Re-send the SafetyOpen with the same configuration, but with incorrect TUNID
7. Confirm the DUT responds with error 01 03, TUNID mismatch
8. Re-send the SafetyOpen with the same configuration, but with another OUNID
9. Confirm the DUT responds with error 01 105, CFUNID mismatch
10. Re-send the SafetyOpen with the same configuration, valid TUNID, OUNID, but with an invalid SCCRC



11. Confirm the DUT responds with error 01 x80C, SCID mismatch
12. Re-send the SafetyOpen with the same configuration, valid TUNID, OUNID, SCID, and CPCRC with a bad connection path
13. Confirm the DUT responds with error 01 117, Connection Path Error
14. Re-send the SafetyOpen with NEW configuration and SCID, but the original OUNID
15. Confirm that the SafetyOpen is accepted and the DUT adopts the new configuration.
16. Confirm the safety I/O connection gets established
17. Send a Type 2b SafetyOpen (null SCID) with a different OUNID to the same output assembly
18. Confirm the DUT sends an error 0x01, 0x106 to indicate OUNID mismatch

Vary the OUNID used in step 6 by changing MAC ID/IP Address and/or SNN to make sure there is no sensitivity to either.

## F-3.10.2 Originators

### F-3.10.2.1 Originators Configuring Targets

The DUT in this section will be an **originator** (sender) of connection requests. The tests called out here will determine if the client is properly issuing the Type 1 Safety Open message. These tests will be highly dependent upon the capabilities and configuration of the client device. For example, not all clients will support configuring a device as part of connection establishment. The client device may also need to be re-configured a number of times to verify all the behavior. The tests should be written in such a way as to allow for developers/testers to isolate or select the test functions that apply to the product's capabilities.

#### F-3.10.2.1.1 TST45 - Type 1 Configuration from Originators

Requirement Number	Requirement
FRS108	<p>If a device either responds to a Type 2 SafetyOpen with a SCID mis-match or requests a download because it is being replaced, the following sequence shall be followed.</p> <p>The originator sends a Type 2 SafetyOpen to the target device</p> <p>The target responds with a mode state error</p> <p>The originator calls the Safety Application to perform a download</p> <p>The Safety Application performs and verifies the download.</p> <p>The download may be a series of commands to the device or</p> <p>The download may be a Type 1 SafetyOpen.</p> <p>The originator repeats the Type 2 SafetyOpen to establish the connection verify the configuration.</p>
FRS110	The Safety Configuration Data CRC (SCCRC) shall only be calculated over the application data that is stored in the device. Any headers or sizes included in the SafetyOpen shall be excluded.
FRS158	The SCCRC itself shall be included in the calculation of the CPCRC
FRS160	The SCTS parameter in the Safety Open shall be included in the calculation of the CPCRC.
FRS172	On DeviceNet, the originator shall establish an explicit messaging connection with the router or end-node to deliver the Forward_Open or SafetyOpen service requests.
FRS170	DeviceNet originators shall use the Connection Object's Safety Open request for local targets
SRS172	If a connection has configuration data, this configuration data shall only be sent when it is



Requirement Number	Requirement
	needed. Unless the configuration has been changed, this shall be accomplished by sending a Type 2 SafetyOpen, and only followed with a Type 1 SafetyOpen if the target responds with Device Not Configured error.
SRS174	When an originator detects that the configuration data held for a device has changed, it shall always issue a type 1 connection request the first time it connects. The SCID returned shall be validated against the value held.

TST45 The Type 1 Configuration Process Test shall confirm a number of requirements required with Type 1 SafetyOpens.

**Initial Conditions:**

1. This test supports both the base format and the Extended Format. The Extended Format will use the EF Safety Segment type
2. Run the SPTE as an originator of either the EF or Base format connections

The following procedure will be used to test an originator DUT type 1 configuration process. This test will be a local subnet test:

1. Configure the originator DUT with a device configuration file and representative connection parameter set
2. Activate the DUT so it attempts to establish a connection
3. Confirm the DUT establishes an explicit connection
4. Confirm the first connection attempt is a Type 1 or a Type 2a connection.
5. Type 1 connection attempt:
6. Check the configuration and calculate the SCCRC over the data.
7. Confirm it matches the value sent.
8. Send a success reply with SCID and confirm the DUT and software recognize successful initial download.
9. Close all connections and power cycle the DUT.
10. Type 2a connection attempt:
11. Send a negative reply with SCID mismatch error code
12. Confirm the connection attempt is a Type 1 connection.
13. Check the configuration and calculate the SCCRC over the data.
14. Confirm it matches the value sent.
15. Send a success reply with SCID and confirm the DUT and software recognize the successful initial download.
16. Close all connections.
17. Confirm the connection attempt is a Type 2a connection.
18. Send a negative reply with SCID mismatch error code
19. Confirm the connection attempt is a Type 1 connection:
20. Check the configuration and calculate the SCCRC over the data.
21. Confirm it matches the value sent.



22. Send a success reply with SCID and confirm the DUT and software recognize the successful initial download.
23. Close all connections
24. Confirm the connection attempt is a Type 2a connection.
25. Send a negative reply with SCID mismatch error code
26. Confirm the connection attempt is a Type 1 connection.
27. Check the configuration and calculate the SCCRC over the data.
28. Confirm it matches the value sent.
29. Send a success reply with SCID and confirm the DUT and software recognize the successful initial download.
30. Close all connections

### **F-3.10.2.2 Connection Configuration Object Tests**

#### **F-3.10.2.2.1 TST105 – Connection Enable/Disable Test**

<b>Requirement Number</b>	<b>Requirement</b>
FRS334	The SafetyClose Service shall be used to close connections with a Safety Targets (and in the case of a ForwardClose all other nodes in the connection path)
FRS339	When a SafetyClose success reply is received, the originator and each intermediate node along the path, closes the connection and releases resources associated with that connection. But the originator shall also close the connection if no response is received within a vendor selected wait period
FRS356	When the Connection Enable/Disable attribute in the CCO object is TRUE, the connection shall remain disabled through power cycles. When the Connection Disable attribute in the CCO object is FALSE, the connection shall remain enabled through power cycles.
FRS357	Safety Devices shall support the open, close and stop services. . These services shall only be accepted when the device is unlocked. If the device is locked, it shall respond with a "Device state conflict" error.

TST105 The Connection Enable/Disable Test shall confirm that the originator DUT maintains the Disabled state through power cycles and the SafetyClose Service closes connections.

#### **Initial conditions:**

1. Commission the DUT with a configuration which has 1 single-cast connection to an I/O device. DUT is initially Unlocked
2. Prompt tester for CCO instance which contains the connection information

#### **The following procedure will be followed for this test:**

1. Allow the DUT to establish those I/O connections
2. Open a Class 3 connection to the DUT
3. Send a Close Connection service to CCO class, instance (provided by tester)
4. Confirm DUT sends a Safety Close service
5. Power cycle the DUT
6. Allow the DUT to establish I/O Connections
7. Confirm DUT does not issue a SafetyOpen to the configured I/O connection
8. Open a Class 3 connection to the DUT



9. Send an Open Connection service to CCO class, instance (provided by tester)
10. Confirm DUT sends a SafetyOpen service
11. Send a Stop Connection service to CCO class, instance (provided by tester)
12. Confirm DUT stops producing data on the connection
13. Power cycle the DUT
14. Allow the DUT to establish I/O Connections
15. Confirm DUT does not issue a SafetyOpen to the configured I/O connection
16. Send an Open Connection service to CCO class, instance (provided by tester)
17. Confirm DUT sends a SafetyOpen service
18. Allow the DUT to establish I/O Connections
19. Send a Close Connection service to CCO class, instance (provided by tester)
20. Confirm DUT sends a Safety Close service
21. Do not send the Safety Close Response
22. Confirm that the DUT shuts down the connection anyway.
23. Open a Class 3 connection to the DUT
24. Send a Lock service to the DUT Safety Supervisor
25. Confirm the DUT sends a success response
26. Send a Close Connection service to CCO class, instance (provided by tester)
27. Confirm DUT sends a “Device State Conflict” error response
28. Send an Open Connection service to CCO class, instance (provided by tester)
29. Confirm DUT sends a “Device State Conflict” error response
30. Send a Stop Connection service to CCO class, instance (provided by tester)
31. Confirm DUT “Device State Conflict” error response

### **F-3.11 SNCT Interface Tests**

The SNCT Interface tests will be a suite of tests covering the objects and behaviors associated with the SNCT interface defined by the System Spec. These tests of course only apply to those device which implement this interface.

#### **F-3.11.1 TST48 - Identity Object Tests**

Requirement Number	Requirement
SRS61	The Identity Object Reset service shall not be supported in safety devices (reply to any requests with “Service Not Supported” error) The service shall be replaced by the Reset service defined by the Safety Supervisor.

**TST48** The Identity Object Tests for safety devices shall confirm that the device has implemented the required behavior changes to the Identity object.

**The following procedure will be rolled into the standard Identity object test:**

1. Open a Class 3 connection with the DUT
2. Issue a proper Reset command to the Identity object
3. Confirm the DUT rejects the message and returns “Service Not Supported” error.



**F-3.11.2 TST50 – MacId, SNN, and UNID Behavior Tests**

<b>Requirement Number</b>	<b>Requirement</b>
SRS106	The MACID attribute (if supported) in the DeviceNet object or the IP attribute in the TCPI/IP object shall always reflect the value in use
SRS107	The Target UNID attribute in the Safety Supervisor shall always reflect either the default or the current Target UNID saved through the UNID setting process.
SRS119	If the UNID represents a valid Number (not equal to out-of-box), this number shall be compared to the MACID or IP Address. In case of a match the device continues its startup procedure. In case of a mismatch the Device transits to Abort.
SRS120	When a device faults from a Switch-NVS setting mismatch, it shall allow either a reset to out-of-box, or a Recover request issued to the Safety Supervisor to clear the Abort. In both cases, the device shall move to the “Waiting for TUNID” state.
FRS350	Safety devices shall monitor and detect changes in the MACID or IP address and Baud Rate (if applicable) switch settings
FRS351	When a MacID switch setting change is detected, the device shall update the MACID_Switch_Value attribute. If the Mac ID switch equals the Mac ID in use or the Mac ID Switches are greater than 63 and software settable clear the Mac ID changed attribute and make no change in the state of the device, otherwise set the Mac ID changed attribute and transition to the ABORT state
FRS352	If a CIP safety device supports the Set_attribute service to the MacId attribute, and the MacId switches are set to enable the set service, Safety devices shall only accept the Set_Attribute while in the “Waiting for TUNID” state. Otherwise, it shall reply with a “Device State Conflict”.
FRS353	If a CIP safety device supports the Set_attribute service to the Baud Rate attribute, Safety devices shall only accept the Set_Attribute while in the “Waiting for TUNID” state. Otherwise, it shall reply with a “Device State Conflict”.
FRS354	When a Baud Rate switch setting change is detected, the device shall update the Baud_Rate_Switch_Changed attribute and the Baud_Rate_Switch_Value attribute, then transition to ABORT.

TST50 The MacId, SNN, UNID and Switch Behavior Test shall confirm that devices behave as required with or without switches.

1. DUT is reset to out-of-box condition

**The following procedure will be followed for this test:**

1. Set switches or MacId attribute to known value
2. Confirm that the Safety Supervisor TUNID value is default value all 0xFF.
3. Execute Propose/Apply to set TUNID
4. Reset DUT to simulate power cycle to confirm the DUT powers up without fault and will accept a configuration (SRS106)
5. Change MacId to different value (switches or MacId attribute) if MacId attribute is set, error Device State Conflict is returned (FRS352)
6. Confirm that the Safety Supervisor TUNID value is the same as in the Apply TUNID. (SRS107)
7. Change BaudRate to different value (switches or BaudRate attribute) if BaudRate attribute is set, error Device State Conflict is returned (FRS353)
8. Reset DUT to simulate power cycle
9. Confirm DUT enters Abort, if MacId attribute is set, state = Idle (SRS119)



10. Send DUT Reset to out-of-box (using original Tunid in reset request)
11. re-establish connection
12. Confirm that Tunid = default (all 0xff)
13. Confirm DUT responds with Waiting for TUNID
14. Set switches to starting value (if soft set, no change occurred)
15. If MacId is switch set, change MacId to different value (NO power cycle, remain ON-LINE)
16. Confirm state = Abort (FRS350) (FRS351)
17. If MacId is switch set, confirm that MacId Switch Changed Attribute = True and MacId Switch Value Attribute = switch value. (FRS350) (FRS351)
18. Change MacId to previous value (NO power cycle, remain ON-LINE)
19. If MacId is switch set, confirm that MacId Switch Changed Attribute = False and MacId Switch Value Attribute = switch value = MACId. (FRS350) (FRS351)
20. If BaudRate is switch set, (do steps 21 through 28, else skip to 29)
21. DUT is reset to out-of-box condition
22. Execute Propose/Apply to set TUNID
23. Change BaudRate to different value (NO power cycle, remain ON-LINE)
24. Confirm state = Abort (FRS350) (FRS354)
25. If Baud Rate is switch set, confirm that Baud Rate Switch Changed Attribute = True and Baud Rate Switch Value Attribute = switch value. (FRS350) (FRS354)
26. Change BaudRate to previous value (NO power cycle, remain ON-LINE)
27. If Baud Rate is switch set, confirm that Baud Rate Switch Changed Attribute = False and Baud Rate Switch Value Attribute = switch value = Baud Rate. (FRS350) (FRS354)
28. Reset to out-of-box (using actual Tunid in reset request) (SRS120)
29. If Baud Rate is software set, (do steps 30 through 35, else skip to 37)
30. DUT is reset to out-of-box condition
31. Execute Propose/Apply to set TUNID
32. Set BaudRate to different value, confirm device response Device\_State\_Conflict (FRS353)
33. Confirm that Safety Supervisor state = Configuring
34. Reset to out-of-box condition (using actual Tunid in reset request) (SRS120)
35. Reset to out-of-box (using default Tunid in reset request) (SRS120)
36. Confirm DUT responds with Waiting for TUNID
37. Confirm that Tunid = default (all 0xff)
38. Configure the DUT

### **F-3.11.3 TST108– IP address switches, SNN, and UNID Behavior**

<b>Requirement Number</b>	<b>Requirement</b>
SRS107	The Target UNID attribute in the Safety Supervisor shall always reflect either the default or the current Target UNID saved through the UNID setting process.
SRS119	If the UNID represents a valid Number (not equal to out-of-box), this number shall be



Requirement Number	Requirement
	compared to the MAC ID or IP Address. In case of a match the device continues its startup procedure. In case of a mismatch the Device transits to Abort.
SRS120	When a device faults from a Switch-NVS setting mismatch, it shall allow either a reset to out-of-box, or a Recover request issued to the Safety Supervisor to clear the Abort. In both cases, the device shall move to the "Waiting for TUNID" state.
FRS350	Safety devices shall monitor and detect changes in the MACID or the IP Attribute and Baud Rate (if applicable) switch settings
FRS352	If a CIP safety device supports the Set_attribute service to the MACID or IP address attribute, and the MACID or IP address switches are set to enable the set service, Safety devices shall only accept the Set_Attribute while in the "Waiting for TUNID" state. Otherwise, it shall reply with a "Device State Conflict".
FRS364	When an IP switch setting change is detected it shall determine if the change creates a mismatch condition. If the IP switch differs from the IP Address attribute, it shall transition to the ABORT state.

TST108 The IP address, SNN, UNID and Switch Behavior Test shall confirm that devices behave as required with or without switches.

1. DUT is reset to out-of-box condition

**The following procedure will be followed for this test:**

1. Set switches or MacId attribute to known value
2. Confirm that the Safety Supervisor TUNID value is default value all 0xFF.
3. Execute Propose/Apply to set TUNID
4. Reset DUT to simulate power cycle to confirm the DUT powers up without fault and will accept a configuration (SRS106)
5. Change MacId to different value (switches or MacId attribute) if MacId attribute is set, error Device State Conflict is returned (FRS352)
6. Reset DUT to simulate power cycle
7. Confirm DUT enters Abort. If MacId attribute is set, confirm state = Idle (SRS119)
8. Confirm that the Safety Supervisor TUNID value is the same as in the Apply TUNID (SRS107)
9. Send DUT Reset to out-of-box (using original Tunid in reset request)
10. Re-establish connection
11. Confirm that Tunid = default (all 0xFF)
12. Confirm DUT responds with Waiting for TUNID
13. Set switches to starting value (if soft set, no change occurred)
14. If MacId is switch set, change MacId to different value (NO power cycle, remain ON-LINE)
15. Confirm state = Abort (FRS350) (FRS351)
16. Change MacId to previous value (NO power cycle, remain ON-LINE)
17. Reset to out-of-box (using default Tunid in reset request) (SRS120)
18. Confirm DUT responds with Waiting for TUNID
19. Confirm that Tunid = default (all 0xFF)



20. Configure the DUT

**F-3.11.4 TST51 - Reset Switch Test**

Requirement Number	Requirement
SRS126	If a reset switch is provided on a safety device, the safety device shall execute a Safety Supervisor Type 2 reset (i.e. reset to out-of-box, but preserve the password).
SRS129	To clear an existing OUNID, safety targets shall support a “reset to out-of-box” using a reset switch on the device or using the special reset command defined in the Safety Supervisor.

TST51 The Reset Switch test shall confirm that the type of reset performed is the equivalent of the Safety Supervisor Type 2 reset .

**The following procedure will be followed for this test:**

1. Open Explicit Message Connection.
2. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.  
**Pass:** Success\_Response, state = ?, IDLE or EXECUTING
3. Take operator intervention to activate the Safety Reset Switch.
4. Open Explicit Message Connection.
5. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.  
**Pass:** Success\_Response, state = 8, waiting for TUNID.
6. Request a Get\_Attribute\_Single, Safety\_Supervisor, Output Connection Owner.  
**Pass:** Success\_Response, OUNID = 0.
7. Request a Safety\_Reset, type = 0, with old Password.  
**Pass:** Success\_Response

**F-3.11.5 Safety Supervisor Functions**

**F-3.11.5.1 TST104 - Baseline Supervisor Test**

Requirement Number	Requirement
SRS65	All safety devices on a CIP safety network shall support the baseline Safety Supervisor definition.

TST104 The Baseline Supervisor Test shall confirm that the device implements all attributes and services marked “required” in the Safety Supervisor object definition.

TBD

**F-3.11.5.2 TST52 - Propose/Apply TUNID and Reset Command Tests**

Requirement Number	Requirement
SRS15	If the device has a Safety I/O connection when a reset command is received, the reset command shall be rejected and an error message returned, “Invalid Mode/State”.
SRS27	The SNCT interface shall provide a special Reset service which requires the password and TUNID to execute and supports the 2 common Reset types along with one that can reset to



<b>Requirement Number</b>	<b>Requirement</b>
	the default but preserve the password
SRS65	All safety devices on a CIP safety network shall support the baseline Safety Supervisor definition.
SRS115	If a device has no switches it shall use default values for MAC Id in its out-of-box configuration (63)
SRS116	The UNID of a device shall be stored to NVS as part of the UNID setting operation.
SRS117	A device in Manufacturers default state shall have an invalid UNID value in its NVS (e.g. 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF).
SRS118	The UNID of a device shall be read from NVS during power up and loaded into an attribute of the Safety Supervisor object and Safety_Network_Number attribute of the DeviceNet, TCP/IP or SERCOS III object.
SRS129	To clear an existing OUNID, safety targets shall support a “reset to out-of-box” using a reset switch on the device or using the special reset command defined in the Safety Supervisor.
SRS134	The default for the Configuration UNID shall be all octets set to 0x00
SRS146	If a TUNID mismatch occurs when sending the Safety Supervisor Safety Reset command, the Invalid Parameter error code (0x20) error code shall be returned
SRS150	If a password mismatch occurs when processing a Safety Supervisor Safety Reset command, a Privilege Violation error code (0x0F) shall be returned
SRS184	The default for SCID attribute in the Safety Supervisor shall be 0
SRS191	The UNID of a device shall be stored to the DeviceNet, TCP/IP or SERCOS III object's SNN attribute as part of the UNID setting operation
SRS194	The Propose_TUNID service shall only be accepted when the target device is in the Waiting_for_TUNID state.
SRS195	The MacId portion of the TUNID shall be matched up against the MacId attribute of the DeviceNet object.
SRS196	If an originator wants to cancel a propose/apply operation, sending a propose service with a TUNID of all 0xFFs shall cancel the operation.
SRS197	The Apply_TUNID service shall validate the TUNID against the proposed value, stops the LED flashing, saves the TUNID to nonvolatile storage, updates the TUNID attribute in the Supervisor, and sets the proposed_TUNID attribute to all 0xFFs. The Supervisor shall then transition to Configuring.
SRS201	The default for all OCPUNIDs shall be 0 (accept any owner) with the ePath for each pointing to the respective resource.

TST52 The Reset Command Test shall confirm the appropriate behavior for handling Safety Supervisor Reset Commands.

**The following procedure will be followed for this test:**

1. Open Explicit Message Connection.
2. Request a Safety\_Open type = 2, SCID = 0, RPI = (100 ms)  
**Pass:** Success\_Response
3. Complete at least one time coordinated handshake to completely establish the connection
4. Request a Safety\_Reset, type = 0.  
**Pass:** Error\_Response, 94 0C FF, Object State Conflict
5. Request a Safety\_Close



**Pass:** Success\_Response

6. Request a Safety\_Reset, with invalid TUNID.

**Pass:** Error\_Response, 94 20 FF, Invalid Parameter.

7. Request a Safety\_Reset, with invalid Password.

**Pass:** Error\_Response, 94 0F FF, Privilege Violation.

## **Type 2 Reset test**

### **Bit Map = 0, retain nothing**

8. Request a Set\_Password, with default Password and new password.

**Pass:** Success\_Response

9. Request a Safety\_Reset, type = 2, bit map = 0, with new Password.

**Pass:** Success\_Response

10. Open Explicit Message Connection.

11. If MacId is settable, Restore the DUT MacId to previous value.

12. Request a Get\_Attribute\_Single, Safety\_Supervisor, CFUNID.

**Pass:** Success\_Response, CFUNID = default, all zero.

13. Request a Get\_Attribute\_Single, Safety\_Supervisor, TUNID.

**Pass:** Success\_Response, TUNID = default, all 0xff.

14. Request a Get\_Attribute\_Single, Safety\_Supervisor, OPCUNID.

**Pass:** Success\_Response, OPCUNID = default, all zero for UNIDs.

15. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.

**Pass:** Success\_Response, state = 8, waiting for TUNID.

### **Bit Map = 1, retain MacId**

16. Request a Set\_Password, with default Password and new password.

**Pass:** Success\_Response

17. Request a Safety\_Reset, type = 2, bit map = 1, with new Password.

**Pass:** Success\_Response

18. Open Explicit Message Connection.

19. Request a Get\_Attribute\_Single, Safety\_Supervisor, CFUNID.

**Pass:** Success\_Response, CFUNID = default, all zero.

20. Request a Get\_Attribute\_Single, Safety\_Supervisor, TUNID.

**Pass:** Success\_Response, TUNID = default, all 0xff.



21. Request a Get\_Attribute\_Single, Safety\_Supervisor, OPCUNID.  
**Pass:** Success\_Response, OPCUNID = default, all zero for UNIDs.
22. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.  
**Pass:** Success\_Response, state = 8, waiting for TUNID.
23. Configure the DUT.

**Bit Map = 2, retain Baud Rate**

24. Request a Set\_Password, with default Password and new password.  
**Pass:** Success\_Response
25. Request a Safety\_Reset, type = 2, bit map = 2, with new Password.  
**Pass:** Success\_Response
26. Open Explicit Message Connection.
27. If MacId is settable, Restore the DUT MacId to previous value.
28. Request a Get\_Attribute\_Single, Safety\_Supervisor, CFUNID.  
**Pass:** Success\_Response, CFUNID = default, all zero.
29. Request a Get\_Attribute\_Single, Safety\_Supervisor, TUNID.  
**Pass:** Success\_Response, TUNID = default, all 0xff.
30. Request a Get\_Attribute\_Single, Safety\_Supervisor, OPCUNID.  
**Pass:** Success\_Response, OPCUNID = default, all zero for UNIDs.
31. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.  
**Pass:** Success\_Response, state = 8, waiting for TUNID.
32. Configure the DUT.

**Bit Map = 5, retain TUNID and MacId**

33. Request a Set\_Password, with default Password and new password.  
**Pass:** Success\_Response
34. Request a Safety\_Reset, type = 2, bit map = 4, with new Password.  
**Pass:** Success\_Response
35. Open Explicit Message Connection.
36. If MacId is settable, Restore the DUT MacId to previous value.
37. Request a Get\_Attribute\_Single, Safety\_Supervisor, CFUNID.  
**Pass:** Success\_Response, CFUNID = default, all zero.
38. Request a Get\_Attribute\_Single, Safety\_Supervisor, TUNID.  
**Pass:** Success\_Response, TUNID = previous non-default value.



39. Request a Get\_Attribute\_Single, Safety\_Supervisor, OPCUNID.  
**Pass:** Success\_Response, OPCUNID = default, all zero for UNIDs.
  40. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.  
**Pass:** Success\_Response, state = 7, Configuring.
  41. Configure the DUT.
- Bit Map = 8, retain Password**
42. Request a Set\_Password, with default Password and new password.  
**Pass:** Success\_Response
  43. Request a Safety\_Reset, type = 2, bit map = 8, with new Password.  
**Pass:** Success\_Response
  44. Open Explicit Message Connection.
  45. If MacId is settable, Restore the DUT MacId to previous value.
  46. Request a Get\_Attribute\_Single, Safety\_Supervisor, CFUNID.  
**Pass:** Success\_Response, CFUNID = default, all zero.
  47. Request a Get\_Attribute\_Single, Safety\_Supervisor, TUNID.  
**Pass:** Success\_Response, TUNID = default, all 0xff.
  48. Request a Get\_Attribute\_Single, Safety\_Supervisor, OPCUNID.  
**Pass:** Success\_Response, OPCUNID = default, all zero for UNIDs.
  49. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.  
**Pass:** Success\_Response, state = 8, waiting for TUNID.
  50. Configure the DUT.

**Bit Map = 16, retain CFUNID**

51. Password is not set, use new password in Reset request
52. Request a Safety\_Reset, type = 2, bit map = 8, with new Password.  
**Pass:** Success\_Response
53. Open Explicit Message Connection.
54. If MacId is settable, Restore the DUT MacId to previous value.
55. Request a Get\_Attribute\_Single, Safety\_Supervisor, CFUNID.  
**Pass:** Success\_Response, CFUNID = previous non-default value.
56. Request a Get\_Attribute\_Single, Safety\_Supervisor, TUNID.  
**Pass:** Success\_Response, TUNID = default, all 0xff.
57. Request a Get\_Attribute\_Single, Safety\_Supervisor, OPCUNID.



**Pass:** Success\_Response, OPCUNID = default, all zero for UNIDs.

58. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.

**Pass:** Success\_Response, state = 8, waiting for TUNID.

59. Configure the DUT.

**Bit Map = 32, retain OCPUNID**

60. Send Type2b Safety Open request (to set OCPUNID).

**Pass:** Success\_Response

61. Request a Set\_Password, with default Password and new password.

**Pass:** Success\_Response

62. Request a Safety\_Reset, type = 2, bit map = 0, with new Password.

**Pass:** Success\_Response

63. Open Explicit Message Connection.

64. If MacId is settable, Restore the DUT MacId to previous value.

65. Request a Get\_Attribute\_Single, Safety\_Supervisor, CFGUNID.

**Pass:** Success\_Response, CFGUNID = default, all zero.

66. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.

**Pass:** Success\_Response, TUNID = default, all 0xff.

67. Request a Get\_Attribute\_Single, Safety\_Supervisor, OPCUNID.

**Pass:** Success\_Response, OPCUNID = previous non-default value.

68. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.

**Pass:** Success\_Response, state = 8, waiting for TUNID.

69. Configure the DUT.

**Bit Map = 63, retain all attributes**

70. Request a Set\_Password, with default Password and new password.

**Pass:** Success\_Response

71. Request a Safety\_Reset, type = 2, bit map = 0, with new Password.

**Pass:** Success\_Response

72. Open Explicit Message Connection.

73. Request a Get\_Attribute\_Single, Safety\_Supervisor, CFGUNID.

**Pass:** Success\_Response, CFGUNID = previous non-default value.

74. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.

**Pass:** Success\_Response, TUNID = previous non-default value.



75. Request a Get\_Attribute\_Single, Safety\_Supervisor, OPCUNID.  
**Pass:** Success\_Response, OPCUNID = previous non-default value.

76. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.  
**Pass:** Success\_Response, state = 7, configuring.

77. Configure the DUT.

**Type 1 reset test**

78. Request a Safety\_Reset, type = 1, with valid Password.  
**Pass:** Success\_Response

79. Open Explicit Message Connection.

80. Request a Get\_Attribute\_Single, Safety\_Supervisor, Output Connection Owner.  
**Pass:** Success\_Response, OUNID = 0.

81. Request a Get\_Attribute\_Single, Safety\_Supervisor, SCID  
**Pass:** Success\_Response, SCID = 0.

----- Propose/Apply TUNID Test -----

Establish initial conditions

82. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.  
**Pass:** Success\_Response, state = 8, waiting for TUNID.

83. Request a Get\_Attribute\_Single, Safety\_Supervisor, Proposed TUNID.  
**Pass:** Success\_Response, value = 0xFFFFFFFF.

84. Request a Get\_Attribute\_Single, Safety\_Supervisor, TUNID.  
**Pass:** Success\_Response, value = 0xFFFFFFFF.

Test detection of nonmatching MacId in TUNID

85. Request a Propose\_TUNID, TUIND = invalid MacId.  
**Pass:** Error\_Response 94 20 FF, Invalid Parameter

86. Request a Propose\_TUNID, TUIND = valid.  
**Pass:** Success\_Response

87. Request Tester confirm the NET LED Flashing at proper rate  
**Pass:** Positive visual confirmation

Test cancellation of proposal

88. Request a Propose\_TUNID, TUIND = 0xFFFFFFFF.



**Pass:** Success\_Response

89. Request Tester confirm the NET LED **stops** flashing

**Pass:** Positive visual confirmation

90. Request a Get\_Attribute\_Single, Safety\_Supervisor, Proposed TUNID.

**Pass:** Success\_Response, value = 0xFFFFFFFF.

91. Request a Get\_Attribute\_Single, Safety\_Supervisor, TUNID.

**Pass:** Success\_Response, value = 0xFFFFFFFF.

Test insensitivity to more than one proposal

92. Request a Propose\_TUNID, TUIND = valid.

**Pass:** Success\_Response

93. Request Tester confirm the NET LED Flashing at proper rate

**Pass:** Positive visual confirmation

94. Request a Get\_Attribute\_Single, Safety\_Supervisor, Proposed TUNID.

**Pass:** Success\_Response, value = from propose request.

95. Request a Get\_Attribute\_Single, Safety\_Supervisor, TUNID.

**Pass:** Success\_Response, value = 0xFFFFFFFF.

96. Request a Propose\_TUNID, TUIND = valid.

**Pass:** Success\_Response

97. Request a Propose\_TUNID, TUIND = Different valid TUNID.

**Pass:** Success\_Response

98. Request a Get\_Attribute\_Single, Safety\_Supervisor, Proposed TUNID.

**Pass:** Success\_Response, value = from propose request.

Test Apply TUNID invalid parameters

99. Request an Apply\_TUNID, TUIND = invalid.

**Pass:** Error\_Response 94 20 FF, Invalid Parameter

100. Request an Apply\_TUNID, TUIND = valid.

**Pass:** Success\_Response

101. Request a Get\_Attribute\_Single, Safety\_Supervisor, Proposed TUNID.

**Pass:** Success\_Response, value = 0xFFFFFFFF.

Test valid Apply TUNID



102. Request a Get\_Attribute\_Single, Safety\_Supervisor, TUNID.

**Pass:** Success\_Response, value = TUNID.

103. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.

**Pass:** Success\_Response, state = 7, configuring.

104. Prompt tester to power cycle device

105. Open Explicit connection

106. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.

**Pass:** Success\_Response, state = 7, configuring.

----- Propose/Apply\_TUNID Test complete -----

107. Request a Configure\_Request, Password = default (all zeros).

**Pass:** Success\_Response

108. Configure the DUT.

109. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.

**Pass:** Success\_Response, state = 2, idle.

110. Request a Safety\_Reset, type = 0, with valid Password.

**Pass:** Success\_Response

111. Open Explicit Message Connection.

112. Request a Get\_Attribute\_Single, Safety\_Supervisor, State.

**Pass:** Success\_Response, state = 2, idle.

### **F-3.11.5.3 TST53 - Configuration Lock/Unlock Test**

<b>Requirement Number</b>	<b>Requirement</b>
SRS10	If an originator attempts to configure a device with the Configuration Lock attribute set, the device shall reject the SafetyOpen and return an error message "Configuration Not Allowed" is returned
SRS25	The SNCT interface shall provide a Configuration Lock service which requires the password and TUNID to execute
SRS31	Safety devices which support the SNCT interface shall require that the "Configuration Lock" attribute be cleared before any command to change a safety device to the Configure State (Configure_Request) will be accepted
SRS32	When the Configuration Lock attribute is set, the device shall reject any Type 1 Safety Open messages.
SRS68	The Configuration Lock attribute, when implemented with the SNCT interface, shall be used to mark the device configuration as verified and to lock down the configuration; 0 = Unlocked (unverified and changeable), 1 = Locked (verified and not changeable).
SRS72	When a Configure_Request is received, if the Configuration_Lock attribute is set, a "Configuration Operation Not Allowed" error shall be returned.
SRS132	The SNCT Configuration Lock Attribute shall have a default value of "Unlocked"
SRS145	If a TUNID mismatch occurs in the Safety Supervisor Configuration_Lock/Unlock command, the Invalid Parameter error code (0x020) shall be returned



<b>Requirement Number</b>	<b>Requirement</b>
SRS149	If a password mismatch occurs when processing a Safety Supervisor Configuration_Lock/Unlock command, a Privilege Violation error code (0x0F) shall be returned
SRS199	While a device has the Configuration Lock attribute set, it shall reject Configure_Requests, Type 1 & 2 Safety Reset, and any valid Type 1 SafetyOpen.
SRS200	When a Type 1 or Type 2 Safety Reset is received, if the Configuration_Lock attribute is set, a "Object State Conflict error code (0x0C)" shall be returned

TST53 The Configuration Lock/Unlock test shall confirm that the DUT supports all the Configuration Lock/Unlock requirements.

**The following procedure or something similar should be followed:**

1. Commission the DUT with MacId, reset to Out-of-Box, and set the TUNID
2. Open a Class 3 connection
3. Configure the DUT with a representative configuration. Leave the PW at the default.
4. Send a Get\_attribute to the Supervisor's Configuration Lock attribute
5. Confirm the attribute is equal to 0
6. Open a Class 3 connection
7. Send the Configuration Lock Command to the safety supervisor with an invalid PW
8. Confirm the message is rejected and a Privilege Violation (0x0F) error is returned
9. Send the Configuration Lock Command to the safety supervisor with an invalid TUNID
10. Confirm the message is rejected and an Invalid Parameter (0x20) is returned
11. Send the Configuration Lock Command (=1) to the safety supervisor with an valid PW and TUNID
12. Confirm the message is accepted
13. Send a Get\_Attribute to the Configuration\_Lock attribute
14. Confirm the value is equal to 1
15. Close all connections and power cycle the device
16. Attempt to configure the device via all supported mechanism (i.e. type 1 and/or Tool interface)
17. Confirm the device rejects all Type 1 connections and/or a valid Configure Requests (i.e. valid parameters)
18. Confirm the error code returned is 0x01 extended code 0x80F (configuration operation not allowed)
19. Send a valid Configure\_Request and confirm error code 0x0C, Object state conflict is returned.
20. Send a Type 1 and Type 2 reset command
21. Confirm both requests return error code 0x0C, Object state conflict.
22. Send a Type 0 Reset
23. Confirm the request is accepted
24. Open a Class 3 connection and clear the Configuration Lock



25. Power cycle the device
26. Confirm the DUT Safety Supervisor state = (2) Idle
27. Attempt to configure the device via all supported mechanisms
28. Confirm the device now accepts all configuration attempts

#### **F-3.11.5.4 TST54 - Configure\_Request Test**

<b>Requirement Number</b>	<b>Requirement</b>
SRS24	The SNCT interface shall provide a Configure Request service which requires password, TUNID, and OUNID to execute
SRS33	The configuration mode in safety devices implementing the SNCT interface shall persist through power cycles.
SRS34	Upon entering the configuration mode, devices implementing the SNCT interface shall. Store the configuration mode in NVS Mark the existing configuration invalid Lock out all configuration messages for connections other than the one that set the configuration mode.
SRS35	To enter the configuration mode, devices implementing the SNCT interface shall: confirm User Password in the configuration mode command confirm TUNID in the configuration mode command
SRS73	When the Configure Request command is accepted, the device shall only accept configuration messages to safety relevant objects over "this connection only". Any attempts to write to safety-relevant objects over other connections shall be rejected.
SRS74	If a configuration connection fails for any reason, another Configure_Request shall be received (establishing a new connection source) before configuration changes can be made (even after a power-cycle)
SRS116	The UNID of a device shall be stored to NVS as part of the UNID setting operation.
SRS130	In the SNCT interface, an unowned device shall capture the OUNID as the device owner when the first Configure_Request is received.
SRS136	All safety devices shall support a non-volatile configuration mode, this means that once a device is commanded to enter the configuration mode it shall remain in the configuration mode until the configuration is successfully validated.
SRS143	If a TUNID, OUNID mismatch error occurs when processing a Safety Supervisor Configure Request, an Invalid Parameter (0x20) error code shall be returned
SRS147	If a password mismatch occurs when processing a Safety Supervisor Configure Request, a Privilege Violation (0x0F) error code shall be returned

TST54 The Configure Request Test shall confirm that the connection target enters the configuration mode only when a properly presented Configuration Request is issued with the proper parameters .

**The following procedure will be followed for this test:**

1. Commission the DUT with a MacId
2. Power cycle the device
3. Open a Class 3 connection to an out-of-box DUT (password at default)
4. Send a Get\_Attribute to the TUNID attribute in the Safety Supervisor
5. Confirm that the TUNID matches the configured values for SNN and MacId
6. Send a Configure\_Request service to the Safety Supervisor with an invalid PW
7. Confirm the request is rejected and Privilege violation (0x0F) error is returned
8. Send a Configure\_Request service to the Safety Supervisor with an invalid TUNID



9. Confirm the request is rejected and an Invalid Parameter (0x20) error is returned
10. Send a Configure\_Request service to the Safety Supervisor with a valid PW, TUNID, and non-tool OUNID
11. Confirm the request is accepted
12. Open another Class 3 connection
13. Send a Configure\_Request service to the Safety Supervisor with a valid PW, TUNID but another OUNID
14. Confirm the request is rejected
15. Close connections and power cycle the device
16. Reopen a Class 3 connection to the DUT
17. Confirm the DUT is in the Configuring state
18. Send a Set\_Attribute to an safety relevant parameter
19. Confirm the DUT rejects the message (it needs to get a new Configure Request first)
20. Send a Configure\_Request service to the Safety Supervisor with a valid PW, TUNID, and another OUNID
21. Confirm the request is rejected with an Invalid Parameter (0x20) error is returned
22. Send a Configure\_Request service to the Safety Supervisor with a valid PW, TUNID, and the original OUNID
23. Confirm the request is accepted
24. Complete the DUT configuration
25. Confirm the configuration is accepted

#### **F-3.11.5.5 TST55 - Configuration Process Test**

<b>Requirement Number</b>	<b>Requirement</b>
FRS116	<p>The following Rules shall apply to device configuration.</p> <p>Devices being configured shall not have any active safety connections and shall not be in the executing mode.</p> <p>Once configuration of a device starts, the device shall remain in the non-operational mode until the configuration process is successfully completed. This mode shall be maintained through power cycles.</p> <p>If a device is in an operational mode (e.g. has active connection or is in executing mode), it shall reject any configuration messages.</p>
SRS26	The SNCT interface shall provide a Validate Configuration service which requires the SCID, SCCRC, and the SCTS to execute
SRS36	A receiving SIL3 device shall calculate a SCCRC with SIL3 integrity and compare it to the SCCRC within the SCID
SRS90	The SCCRC shall be calculated over the configuration data by the device whenever a device configuration is validated (i.e. Validation Request to the safety supervisor or type 1 safety connection)
SRS91	The SCCRC shall be saved to NV storage along with the other NV attributes whenever a configuration is applied (i.e. apply request to the safety supervisor).
SRS131	The SNCT interface shall provide an Apply service that causes the device to save the configuration to NV memory
SRS138	Any explicit messages are allowed to set configuration in the device, but the device shall only accept safety-related explicit messages over the connection that put it into the configuration mode.



Requirement Number	Requirement
SRS151	If the Safety Supervisor Validate Configuration command does not succeed, the error response shall send a class defined "Validation Error" (0xD0)
SRS198	When a device enters Configuring mode, the SCID attribute shall be set to 0 and maintained at this value (through power cycles) until a successful Validate has been executed.

TST55 The Configuration Process Test shall confirm that the DUT follows the basic safety configuration procedure dictated by the Safety Supervisor

**The following procedure will be followed for this test:**

1. Open a Class 3 connection to the DUT
2. Send a set attribute service to a safety related configuration object (ie. Configuration assembly, attr. 3)
3. Confirm the DUT responds with a Mode/State error
4. Send a Configure\_Request with the proper parameters (ie. Password, OUNID)
5. Get and confirm that the SCID = 0
6. Send the same Set\_Attribute service as before
7. Confirm the device accepts the message
8. Power cycle the device
9. Open a Class 3 connection with the DUT
10. Get and confirm that the SCID = 0
11. Read the Safety Supervisor state attr and confirm the state is Configuring
12. Send the same Set\_Attribute service
13. Confirm the device rejects the service (you need to resend the Configure\_Request first)
14. Send a Configure\_Request with the proper parameters
15. Send the same Set\_Attribute service as before
16. Confirm the device accepts the message
17. Open another Class 3 connection (as if another configuration device were connecting)
18. Send a Set\_Attribute service to a safety relevant parameter over this new connection
19. Confirm that the set attribute is rejected
20. Send a complete configuration over the original connection
21. Send a Validate Configuration Request with a invalid CRC in SCID
22. Confirm the device sends a Validation Error (0x0D, 01)
23. Send a Validate Configuration Request with a valid SCID
24. Confirm the device sends a success reply
25. Inspect the returned SCID and confirm the SCCRC matches the value sent.
26. Send a Configuration Apply
27. Close connections and power cycle device
28. Confirm the DUT is in the IDLE state and the configuration SCID is correct



### **F-3.11.5.6 TST56 - Setting Passwords Test**

<b>Requirement Number</b>	<b>Requirement</b>
SRS17	Support for one password shall be implemented by devices supporting the SNCT interface
SRS20	The default password value in devices supporting the SNCT interface shall be zero
SRS22	The SNCT interface shall support a Password service to set passwords which takes 2-password parameters; Old PW and New PW
SRS148	If a password mismatch occurs on the current password when processing a Safety Supervisor Set Password command, a Privilege Violation error code (0x0F) shall be returned

TST56 The Setting Password Test shall confirm a number of the password feature requirements are being met.

**The following procedure will be followed:**

1. Request a Safety\_Reset, type = 1.  
**Pass:** Success\_Response
2. Request a Propose\_TUNID, TUIND = valid.  
**Pass:** Success\_Response
3. Request an Apply\_TUNID, TUIND = valid.  
**Pass:** Success\_Response
4. Open an Explicit Message Connection
5. Request the default Configure\_Request, password = default (all zeros).  
**Pass:** Success\_Response
6. Request a Set\_Password, old = not default, new = default.  
**Pass:** Error\_Response, 94 0F FF, Privilege Violation
7. Request a Set\_Password, old = default, new = non-default.  
**Pass:** Success\_Response
8. Request a Safety\_Reset, type = 0.  
**Pass:** Success\_Response
9. Open an Explicit Message Connection.
10. Request a Configure\_Request, password = new password  
**Pass:** Success\_Response
11. Configure the DUT.
12. Request a Validate\_Configuration with valid SCID  
**Pass:** Success\_Response
13. Request a Set Password, old = new password, new = default.



Pass: Success\_Response

### F-3.11.5.7 TST57 - Mode Change Test

Requirement Number	Requirement
SRS28	The SNCT interface defines an optional Mode Change service which, if implemented, shall require a password to execute
SRS144	If a password mismatch occurs in the Safety Supervisor Mode Change command, an Privilege Violation error code (0x0F) shall be returned

TST57 The Mode Change Test shall confirm that devices which support the Mode change also require the proper Password parameter to execute

The following procedure will be followed:

1. Open a Class 3 connection to the DUT
2. Request Type 0 Safety Reset
3. Verify that the Safety Supervisor state = 2, Idle
4. Request Mode\_Change service, state = RUN with invalid password
5. Verify that the error response with general status 0x0F, Privilege Violation
6. Request Mode\_Change service, state = RUN with valid password
7. Verify that the success response
8. Verify that the Safety Supervisor state = 4, Executing
9. Request Mode\_Change service, state = Idle with valid password
10. Verify that the success response
11. Verify that the Safety Supervisor state = 2, Idle

### F-3.11.6 Safety Validator Object Tests

#### F-3.11.6.1 TST58 - Safety Connection Diagnostics

Requirement Number	Requirement
SRS75	This Safety Connection Fault Count attribute in the Safety Validator shall be a running count (with auto-rollover) of how many times any safety connection was faulted for any reason.
SRS78	The Max Data Age attribute shall always reflect the largest value the connection's data age reaches without exceeding the expectation multiplier.
SRS79	If a safety connection is ever faulted and closed, the Max Data Age attribute shall be re-initialized to 0 when the connection is re-established.

TST58 The Safety Connection Diagnostic test shall confirm that the DUT supports the diagnostic attributes and the behavior is correct.

The following procedure will be followed for this test:

1. Commission and configure the DUT with a representative configuration
2. Establish a safety connection with the device
3. Open a Class 3 connection to the DUT
4. Send a Get\_Attribute to Validator class to read the Fault count attribute
5. Send a Get\_Attribute to Validator instance to read the Max Data Age attribute



6. Generate a Safety connection fault by sending bad safety CRCs
7. After the safety connection has closed, send a get attribute to the validator class and confirm the Fault count was incremented.
8. Re-establish the safety connection
9. Generate increasing delays in communication to cause the Data age to increase in the device but within the tolerance of the connection
10. Send a get attribute to the Max Data Age and confirm the value increases as the delay is increased.
11. Cause the safety connection to fault again
12. Once the connection gets closed, reopen it
13. Send a get attribute to the Max Data Age and confirm the value is reset to 0.

### **F-3.12 DeviceNet Bridge Device Tests**

The following tests are provided to aid in the development of standard conformance testing of bridging/router devices which have implemented safety routing functionality. The requirements referenced here are all non-safety-related interoperability requirements.

#### **F-3.12.1 TST62 - DeviceNet Bridge FW\_Open to SafetyOpen Conversion**

<b>Requirement Number</b>	<b>Requirement</b>
Interoperability Requirement only	The last router/bridge in a multi-hop connection shall convert the Forward_Open requests received from a higher-level network into the equivalent Safety Open request when the target is on DeviceNet.

TST62 The DeviceNet Bridge FW\_Open to SafetyOpen Conversion test shall confirm that the DeviceNet bridge DUT will perform the conversion properly.

#### **F-3.12.2 TST63 - DeviceNet Bridge Non-Dnet-to-Dnet Forwarding Test**

<b>Requirement Number</b>	<b>Requirement</b>
Interoperability Requirement only	A non-certified DeviceNet Bridge, bridging a multi-cast production from a Non-DeviceNet network to a DeviceNet network shall forward a Data Message consisting of a Data&Time_Stamp section onto the DeviceNet network, once and only once, any time a Data Message is received from the Non-DeviceNet
Interoperability Requirement only	A non-certified DeviceNet Bridge, bridging a multi-cast production from a Non-DeviceNet network to a DeviceNet network shall forward a Time Correction Message onto the DeviceNet network, once and only once, any time a Data Message is received from the Non-DeviceNet network with a Time Correction section - MCast_Byte.Consumer_Num equal to a value other than 0x0. If the MCast_Byte.Consumer_Num is equal to 0x0 (Null Time_Correction section), the Time Correction Message shall not be forwarded
Interoperability Requirement only	The non-certified DeviceNet Bridge shall not translate a Time_Coordination Message. It is forwarded upon reception.
Interoperability Requirement only	The DeviceNet Bridge shall never create or modify Safety CRCs for the Data&Time_Stamp, Time_Correction, or Time_Coordination sections.

TST63 The DeviceNet Bridge Non-Dnet-to-Dnet Forwarding Test shall confirm that the required rules for forwarding safety Data messages are being followed.



**The following procedure will be followed:**

1. Set up a system with a tester acting as a multi-cast producing target on a non-DeviceNet network and a tester acting as 5 multi-cast originators on DeviceNet.
2. Originate a connection to the target through the bridge DUT
3. Count production data messages sent to the multi-cast consumers
4. Confirm that the number of messages sent equals the number captured by the consuming tester
5. Send a predefined number of data productions with the Mcast\_Byte Consumer Num = 0
6. Confirm the bridge does not produce Time Correction messages
7. Begin sending the data productions with the Mcast Byte Consumer Num with non-zero values
8. Confirm the Time Correction message are forwarded as sent
9. Confirm the CRCs are never corrupted or incorrect at any time during the test

**F-3.12.3 TST64 - DeviceNet Bridge Dnet-to-nonDnet Forwarding Test**

<b>Requirement Number</b>	<b>Requirement</b>
Interoperability Requirement only	When multi-cast connections are bridged between DeviceNet and other Non-DeviceNet networks, the safety-aware DeviceNet bridge shall translate the connections from three messages on DeviceNet to two connections on Non DeviceNet Networks
Interoperability Requirement only	The safety-aware DeviceNet bridge shall execute the 3-to-2 multi-cast safety connection translation by concatenating the safety data message and the time correction message at the bridge.
Interoperability Requirement only	The non-certified DeviceNet Bridge shall not translate a Time_Coordination Message It is forwarded upon reception.
Interoperability Requirement only	A non-certified DeviceNet Bridge, bridging a multi-cast Data Message and Time_Correction Message from a DeviceNet network to a Non DeviceNet network shall send a Data Message consisting of a Data&Time_Stamp section and a Time Correction section onto the Non-DeviceNet network any time it receives a Data Message consisting of a Data&Time_Stamp section from the DeviceNet network.
Interoperability Requirement only	Every Time_Correction Message or Messages received from the DeviceNet network shall be forwarded once and only once in order, in the Time Correction section of the Non-DeviceNet Data Messages that are triggered by the reception of the DeviveNet Data Message
Interoperability Requirement only	When sending a Data Message onto the Non-DeviceNet network, if there are no Time_Correction Messages to be forwarded, the previous Time_Correction section shall be sent with the MCast Byte.Consumer Num equal to 0x0 (Null Time_Correction section)
Interoperability Requirement only	For the case of the first Data Message being sent onto Non-DeviceNet before a DeviceNet Time_Correction Message has been received, a Time_Correction section of all 00's shall be sent.
Interoperability Requirement only	The DeviceNet Bridge shall never create or modify Safety CRCs for the Data&Time_Stamp, Time_Correction, or Time_Coordination sections.

TST64 The DeviceNet Bridge Dnet-to-nonDnet Forward Test shall confirm that the required rules for forwarding safety data messages to non-devicenets networks are followed.



The following procedure will be followed:

1. Set up a system with a tester acting as 5 multi-cast consuming originators on a non-DeviceNet network and a tester acting a multi-cast producer on DeviceNet.
2. Open the connections to the producer through the bridge DUT
3. Confirm that the bridge DUT constructs the proper SafetyOpen with the extra Time Correction connection
4. Send production data without Time Correction messages
5. Confirm the bridge DUT sends message on the non-DeviceNet side with Time Correction section consisting of all 00's. *The CRC for this is irrelevant and should be ignored.*
6. Sending a Time Correction message to one of the consumers
7. Confirm that the the produced data with the correct Time Correction section added (with a proper CRC)
8. Confirm that all Time Correction messages get forwarded only one time and all subsequent production are followed by a Null Time Correction
9. repeat for all the other consumers.
10. Confirm the all data and time stamp section CRCs are always correct

### F-3.13 Standard CIP Message Interaction

#### F-3.13.1 Standard Messaging Rejection

##### F-3.13.1.1 TST65 - Standard messaging to Safety I/O

Requirement Number	Requirement
FRS8	Safety consumer shall detect standard messages as an incorrectly formed message.

TST65 This test shall set up a safety I/O connection of a type supported by the DUT and send a standard slave I/O message over the connection. The test will confirm that the device detects the message as an improper format and reject the message.

## F-4 White Box Tests

White box tests are those executed by the product developer. Because of the lack of adequate external visibility to certain aspects of the protocol, those requirements which are not externally testable will be placed in the White box test area. It will be the responsibility of the product developer to generate appropriate tests to confirm the product is meeting these requirements. It is beyond the scope of this document to define exactly how the requirements are confirmed. Many may be simply done through code inspection or design review.



## **F-4.1 Application Behavior**

### **F-4.1.1 TST61 - Safety Input Application Behavior Test**

<b>Requirement Number</b>	<b>Requirement</b>
FRS117	If a safety data state exists for the safety-input device, the device shall transmit safety data and the Run_Idle bit shall be set to the idle state upon detection of a fault by the input circuit diagnostics.
FRS118	Changes in the state of the physical process inputs of a safety device shall be transmitted at the next available periodic update time.
SRS48	The default safety state for digital inputs and outputs shall be off.

TST61 The Safety Input Run/Idle test shall confirm that input device applications behave properly.

**The following procedure will be followed for this test:**

1. Establish a Safety connection to an Input DUT with a representative configuration
2. Once the safety connection is established, confirm the Run/Idle bit goes to Run
3. Set the physical inputs to a non-safety-state and confirm the produced data reflects the inputs
4. Create a fault condition to cause the DUT to go to safety-state
5. Confirm the Run/Idle bit goes to Idle and the data is set to its safety-state

### **F-4.1.2 TST60 - Safety Output Application Behavior Test**

<b>Requirement Number</b>	<b>Requirement</b>
SRS48	The default safety state for digital inputs and outputs shall be off.
FRS288	If consumed safety data becomes older then Network_Time_Expectation_Multiplier, the safety data shall be put in the safety state

TST60 The Safety Output Application Behavior Test shall confirm that the output device always safety-states to the off condition.

**The following procedure will be followed for this test:**

1. Establish a Safety connection to the Output DUT with a representative configuration
2. Once the safety connection is established, confirm the Run/Idle bit goes to Run and send non-zero output data
3. Confirm the outputs are driven to non-zero states
4. Delay communication until the DUT application detects a data age error
5. Confirm the outputs go to the zero state
6. Re-establish communication
7. Create any and all additional application specific fault conditions to cause the DUT to go to safety-state
8. Confirm the outputs go to the zero state



## **F-4.2 Consumer Behavior**

### **F-4.2.1 TST66 - Consumer Initialization**

<b>Requirement Number</b>	<b>Requirement</b>
FRS278	At connection establishment, all S_Run_Idle_Out flags shall be set to Idle.
FRS292	Since the initial production of safety data should have a Ping_Count of 00, the Last_Ping_Count shall be initialized to 11.
FRS297	When a connection is first established, the Time_Coordination_Count_Down shall be initialized to 0.
FRS300	When a connection is first established, the Last_Data_Time_Stamp shall be initialized to 0
FRS302	When a connection is first established, the Last_Rcvd_Multi_Cast_Active_Idle shall be initialized to 0.
FRS304	When a connection is first established, the Last_Rcvd_Time_Correction_Value shall be initialized to 0.

TST66 The designer shall confirm via inspection or design review that the safety protocol consumer variables are re-initialized on connection establishment

### **F-4.2.2 TST67 - Consumer Mode Byte Processing**

<b>Requirement Number</b>	<b>Requirement</b>										
FRS37	<p>The Base Format Mode byte processing within the CRCs shall be done as shown as follows.</p> <table> <tr> <td>CRC Type</td><td>Mode byte logic</td></tr> <tr> <td>Actual Data CRC</td><td>Mode Byte AND 0xE0</td></tr> <tr> <td>Complement Data CRC</td><td>(Mode Byte XOR 0xFF)</td></tr> <tr> <td></td><td>AND 0xE0</td></tr> <tr> <td>Time Stamp Section CRC</td><td>Mode Byte AND 0x1F</td></tr> </table>	CRC Type	Mode byte logic	Actual Data CRC	Mode Byte AND 0xE0	Complement Data CRC	(Mode Byte XOR 0xFF)		AND 0xE0	Time Stamp Section CRC	Mode Byte AND 0x1F
CRC Type	Mode byte logic										
Actual Data CRC	Mode Byte AND 0xE0										
Complement Data CRC	(Mode Byte XOR 0xFF)										
	AND 0xE0										
Time Stamp Section CRC	Mode Byte AND 0x1F										

TST67 The Designer shall verify that the consumer FW is properly isolating and including the correct Mode Byte sections in the CRC calculations

### **F-4.2.3 TST68 - Consumer Time Value Sampling**

<b>Requirement Number</b>	<b>Requirement</b>
FRS51	Consumer_Time_Value in the Time Coordination message shall (as closely as possible) mark the time at which the consumer sends a ping response message

TST68 The designer shall confirm that the design is sampling the consumer time value as close as possible to the point at which the Time Coordination response is sent.

### **F-4.2.4 TST69 - Consumer Time Coordination Generation**

<b>Requirement Number</b>	<b>Requirement</b>
FRS294	The Time_Coordination_Count_Down shall be set equal to the Consumer_Number upon the reception of a multi-cast ping request.
FRS295	The consumer shall decrement the Time_Coordination_Count_Down value on subsequent reception of consumed data until the value is equal to 0
FRS296	If the Time Coordination Count Down value is equal to 1 during any reception of consumed data, the ping response for that consumer shall be sent



TST69 The Designer shall confirm via inspection or design review that the Time Coordination distribution function provided by the Time Coordination Count Down mechanism is behaving as required for all possible consumer numbers.

**F-4.2.5 TST70 - Consumer Timeout Multiplier Handling**

Requirement Number	Requirement
FRS82	Safety communications shall be configurable with production repeated.
FRS83	When safety communication is sent with repeated production, application-triggered consumers shall use the last packet received to complete the transaction, as show in Figure 25.
FRS88	Valid repeated messages shall be overwritten. The most recent valid message is used by the application

TST70 The designer shall confirm that the device supports the valid range of Timeout Multipliers and uses the latest data when overproduction is used

1. Setup a series of connections with the DUT with Timeout Multipliers equal to 1, 2, 3, and 4.
2. Set the EPI to occur at a rate that exceeds the application sampling rate (this only applies to application triggered consumers)
3. Deliver data with different data each production
4. Confirm that the data issued to the application is always the latest data available
5. Repeat this test for all consumer types supported
6. Confirm that the consumer captures the Time Stamp in the redundant arrivals

**F-4.2.6 TST71 - Multi-Cast Consumer Time Correction Handling**

Requirement Number	Requirement
FRS28	The Multi-cast safety data consumer shall use the Producer_Time_Stamp and the Time_Correction_Value to derive a Data_Time_Stamp that is relative to the safety data consumer's clock
FRS61	Consumer_Time_Correction_Value in the Time Correction message shall be used to correct the data time stamp and make the time relative to the multi-cast consumers clock
FRS286	The Multi-cast consumer shall enable time stamp checking upon the reception of the first time correction section.
FRS298	For multi-cast, the Corrected_Data_Time_Stamp shall be a sum of the received time stamp and the Last_Rcvd_Time_Correction_Value
FRS303	The Last_Rcvd_Time_Correction_Value shall be used to correct the Time Stamp for each multi-cast consumer.
FRS306	When the Time_Correction_Received_Flag is set, the consumer shall begin the process of correcting and checking the time stamp.

TST71 The designer shall perform a test that simulates a series of Time Correction Messages with various Time Correction Values and confirm the design applies these values to adjust the Data Time Stamp. The designer shall confirm that Time stamp checking is held off until the first Time Correction message is received.



#### **F-4.2.7 TST72 - Message Age Failure Indication**

<b>Requirement Number</b>	<b>Requirement</b>
FRS340	Consumer_Clk_Count at the time that the data is given to the consuming application for use and the Data_Time_Stamp sent with the data exceeds the Network_Time_Expectation_Multiplier, the S_Con_Flt_C_Out flag shall be set to "Faulted" to indicate that the Reaction Time for the network has not been met.
FRS284	If the consumer receives the Data_Time_Stamp as a value other than 0, the time stamp checking shall be enabled.

TST72 The designer shall determine via design review or unit test that the data age test is functioning correctly and the application is notified by faulting the connection. Confirm the age checking is held off until a non-zero time stamp is received.

#### **F-4.2.8 Link-Triggered Behavior**

##### **F-4.2.8.1 TST73 - Link-Triggered Consumer Behavior Checks**

<b>Requirement Number</b>	<b>Requirement</b>
FRS127	the SafetyValidatorServer shall perform the following aspects of the safety data monitoring: Time coordination with the producer Time stamp section integrity checking Data integrity checking Network time expectation checking
FRS128	The Link Triggered SafetyValidatorServer shall perform all aspects of the safety protocol on each message received
FRS129	For multi-cast, the Link Triggered SafetyValidatorServer shall also perform all aspects of the safety protocol on each Time Correction section received.
FRS130	The SafetyValidatorServer shall insure that the safety data presented to the consuming application has been crosschecked and that the network time expectation is being met.
FRS131	The Link Triggered SafetyValidatorServer shall set the new data flag each time data is ready for consumption.

TST73 The designer shall confirm that the Link-triggered safety designs performs all the required checks on Time coordination, Time stamps, and Data crosschecking on each reception and signals the application. The designer shall also confirm that multi-cast connection process Time correction messages on reception.

#### **F-4.2.9 Application-Triggered Behavior**

##### **F-4.2.9.1 TST74 & TST75 - Application Triggered Consumer Behavior Checks**

<b>Requirement Number</b>	<b>Requirement</b>
FRS127	the SafetyValidatorServer shall perform the following aspects of the safety data monitoring: Time coordination with the producer Time stamp section integrity checking Data integrity checking Network time expectation checking
FRS132	The process reception stage (Figure 45) of the application triggered SafetyValidatorServer shall check the ping count for all incoming data messages.



<b>Requirement Number</b>	<b>Requirement</b>
FRS133	For multi-cast consumption in application triggered SafetyValidatorServers, the latest Time_Correction section received shall be forwarded to the consuming application
FRS134	The consuming safety application shall request new data from the SafetyValidatorServer in order to cause the SafetyValidatorServer to complete processing of a message
FRS135	The consuming application shall ensure that it gets new data at least once every 2 seconds to ensure that the time stamp does not rollover.

TST74 The designer shall confirm that the application triggered safety designs check the ping count (and consumer # for multi-cast) on arrival and captures the Time Correction sections when the consumer number matches in the multi-cast case.

TST75 The designer will confirm that when the application requests new safety data, stage 2 consumer processing is executed. The designer shall confirm that the application makes this request at once every 2 seconds.

#### **F-4.2.10 TST76 - Data Integrity Check Time Limit**

<b>Requirement Number</b>	<b>Requirement</b>
FRS145	The data integrity shall be checked at least once every 2 seconds.
FRS146	The Time Stamp and Network Time Expectation (NTE) shall be checked at least once every 2 seconds

TST76 The Designer shall confirm via design review or special test that the device design has a mechanism implemented that assures the safety protocol will be executed at least once every 2 seconds.

#### **F-4.2.11 TST77 - Safety Consumer/Application Interface**

<b>Requirement Number</b>	<b>Requirement</b>
FRS34	Run_Idle shall be used to indicate the usability of the data as determined by the Producer Safety Application
FRS276	S_Run_Idle_Out shall be used by a consuming application to determine if it should put the safety data in a safety state.
FRS277	The S_Run_Idle_Out shall indicate Idle when either the producing application is Idle or the Safety Validators are not active
FRS274	The Connection Status of consuming safety connections shall be indicated as shown in Table 2-4.2 of Volume 5 Chapter 2
FRS279	S_Run_Idle_Out shall be in Idle until Init_Complete_Out is set to a 1
FRS301	The Last_Rcvd_Multi_Cast_Active_Idle shall be used to detect a change from active to idle, which would be considered a fault.

TST77 The designer shall confirm via design review of system test that as application uses the S\_Run\_Idle to decide when to consume safety data into the safety state. The designer shall also confirm that once a producer has gone Active, a transition to Idle will cause the connection to fault and the application faulted



### **F-4.3 Producer Behavior**

#### **F-4.3.1 TST78 - Producer Initialization**

<b>Requirement Number</b>	<b>Requirement</b>
FRS263	At connection establishment, all Consumer_Time_Value variables shall be set to 0x0000.
FRS266	At connection establishment, all Producer_Rcvd_Time_Value variables shall be set to 0x0000.
FRS268	At connection establishment, all Consumer_Time_Correction_Value variables shall be set to 0x0000.
FRS273	At connection establishment, all Ping_Int_Since_Last_Time_Coord_Msg_Count variables shall be set to 0x0000
FRS275	At connection establishment, all S_Con_Flt_C_Out flags shall be set to OK.

TST78 The designer shall confirm via inspection or design review that the safety protocol variables are re-initialized on connection establishment.

#### **F-4.3.2 TST79 - Producer Mode Byte Processing**

<b>Requirement Number</b>	<b>Requirement</b>										
FRS37	<p>The Mode byte processing within the CRCs shall be done as shown as follows.</p> <table> <tr> <td>CRC Type</td><td>Mode byte logic</td></tr> <tr> <td>Actual Data CRC</td><td>Mode Byte AND 0xE0</td></tr> <tr> <td>Complement Data CRC</td><td>(Mode Byte XOR 0xFF)</td></tr> <tr> <td></td><td>AND 0xE0</td></tr> <tr> <td>Time Stamp Section CRC</td><td>Mode Byte AND 0x1F</td></tr> </table>	CRC Type	Mode byte logic	Actual Data CRC	Mode Byte AND 0xE0	Complement Data CRC	(Mode Byte XOR 0xFF)		AND 0xE0	Time Stamp Section CRC	Mode Byte AND 0x1F
CRC Type	Mode byte logic										
Actual Data CRC	Mode Byte AND 0xE0										
Complement Data CRC	(Mode Byte XOR 0xFF)										
	AND 0xE0										
Time Stamp Section CRC	Mode Byte AND 0x1F										

TST79 The Designer shall verify that the consumer FW is properly isolating and including the correct Mode Byte sections in the CRC calculations

#### **F-4.3.3 TST80 - Producer Time Coordination Response Handling**

<b>Requirement Number</b>	<b>Requirement</b>
FRS72	The producer shall use the clock count in the Time Coordination response to calculate an offset that will be applied to future productions.
FRS73	At each scheduled production, the producer shall use the value of its own clock count and the offset to calculate the time stamp and sends it to the consumer along with the data.
FRS229	The Timeout_Multiplier shall be used to determine how many ping intervals (Timeout_Multiplier.PI+2) a producer will wait before faulting a consumer who has failed to send a Time Coordination response to a Ping request
FRS260	The Time_Drift_Since_Last_Time_Coord shall be used to indirectly determine if the delay of the Time Coordination message received is significantly greater than the delay for the Time Coordination message previously being used.
FRS261	Since the maximum time drift is controlled, either the Last Time_Correction_Value minus the Time_Drift_Since_Last_Time_Coord value, or the New Time_Correction_Value value, shall be sent to the consumer, whichever is better.
FRS262	For multi-cast producers, a Time_Drift_Since_Last_Time_Coord shall exist for each multi-cast consumer, 1 through Max_Consumer_Number.
FRS264	Producer_Rcvd_Time_Value shall be set equal to Producer_Clk_Count at the point in time that the Time Coordination Message information is received from the time stamp consumer.



Requirement Number	Requirement
FRS265	A Producer Rcved_Time_Value shall exist for each multi-cast consumer, 1 through Max_Consumer_Number.
FRS269	For single-cast connections, the Ping_Int_Since_Last_Time_Coord_Msg_Count shall be incremented at the 8th EPI production of the Ping Interval.
FRS270	For multi-cast connections, Ping_Int_Since_Last_Time_Coord_Msg_Count shall be incremented every $(8 + n)$ th EPI production of the Ping Interval, where n equals the consumer number - 1.
FRS271	The Ping_Int_Since_Last_Time_Coord_Msg_Count for any particular consumer shall be reset upon the reception of a valid Time Coordination Message from that consumer
FRS272	If the Ping_Int_Since_Last_Time_Coord_Msg_Count ever exceeds the Timeout_Multiplier.PI + 2 for a connection, that consumer shall be set to the faulted state
FRS320	If the Consumer Active_Idle[Consumer_Num-1] = Active, Producers shall check that Time Coordination Messages are received within the Time_Coord_Response_EPI_Limit and fault the connection if they are not.
FRS360	When a Time Coordination Message is received; the producer shall check that the Time_Coordination_Section.Consumer_Time_Value is not the same as the last received Consumer_Time_Value. If it is, the Time Coordination message shall not be processed.

TST80 The designer shall confirm that the Time value in the Time Coordination response is being applied to create the proper time offset on future time stamps.

This inspection should be performed for all supported production services (ie. Multi-cast and single-cast).

TST81 The designer shall confirm via design review or inspection that the Time\_Drift\_Since\_Last\_Time\_Coord is being measured and used to in cases to calculate Time Correction Values when needed. Also the designer shall confirm that a drift value is kept for each multi-cast consumer.

TST82 The designer shall confirm via design review or inspection that the Producer\_Rcved\_Time\_Value is implemented as required.

TST83 The Designer shall confirm via design review or inspection that the Ping\_Int\_Since\_Last\_T\_Coord\_Msg\_Count is incremented correctly for single-cast and multi-cast, plus the designer shall confirm it gets reset when Time Coord messages are received. The designer shall also confirm that this parameter is used to fault the connection when the Time Coord Msg is not received within the proper time period.

#### **F-4.3.4 TST85 - Producer Parameter Calculations**

Requirement Number	Requirement
FRS245	The Time_Drift_Constant calculation shall be done using at least 32 bit integer math with the result being a 16 bit integer.
FRS247	The Time_Coord_Response_EPI_Limit calculation shall be done in 32 bit integer math with the result being a 16 bit integer

TST85 The designer shall confirm by inspection or design review that the Time\_Drift\_Constant and Time\_Coord\_Response\_EPI\_Limit calculation is done with the required mathematical accuracy



#### **F-4.3.5 TST86 - Safety Producer/Application Interface**

<b>Requirement Number</b>	<b>Requirement</b>
FRS123	Safety_Data_Out is produced at a periodic rate, referred to as the Expected Packet Interval (EPI). The SafetyValidatorClient shall sample, capture, and time stamp the data to be sent every EPI time period
FRS124	The producing application provides Safety_Data_Out to the SafetyValidatorClient. The SafetyValidatorClient shall build the Mode_Byte, Actual_Data, and Complement_Data, and Time stamp section (see section 5.7.1 for formats).
FRS125	The SafetyValidatorClient shall provide safety connection status back to the producing application for each consumer.
FRS217	The producer connection status shall be determined by the combination of the S_Connection_Fault and the Consumer_Active_Idle flags (Refer to Table 2-4.1)
FRS219	If a safety data producer is producing data and valid Time Coordination information has been received without errors the Consumer_Active_Idle flag shall be equal to Active, otherwise it will be Idle.
FRS220	For multi-cast producers, a Consumer_Active_Idle flag shall exist for each multi-cast consumer, numbered 1 through Max_Consumer_Number.
FRS221	At connection establishment, all Consumer_Active_Idle flags shall be initialized to Idle.
FRS222	An S_Connection_Fault indication of Fault, shall indicate that the producer has detected a problem with the safety connection to this consumer or the consumer has indicated an error back to the producer
FRS223	If the safety data producer has received invalid Time Coordination information, Time Coordination information with a Con_Detected_Fault set to Faulted, or a Time Coordination information timeout, the S_Connection_Fault flag shall be equal to Fault, otherwise it will be OK
FRS224	For multi cast producers, a S_Connection_Fault flag shall exist for each multi cast consumer, numbered 1 through Max_Consumer_Number.
FRS225	At connection establishment, all S_Connection_Fault variables shall be initialized to OK
FRS250	Producer_Safe_Data_TS is a variable that shall be set equal to Producer_Clk_Count at the point in time that the Safety_Data for a particular EPI data production is sampled and captured from the producing application

TST86 The Designer shall confirm that the Safety Producer captures the latest data for each EPI production and provides timely consumer connection status back to the application. The designer shall confirm that the captured data is reflected in the true and complement data sections, and the status is derived properly.

#### **F-4.3.6 TST87 - Connection Establishment – RunTime Coordination**

<b>Requirement Number</b>	<b>Requirement</b>
FRS322	For Safety Data producers, Consumer_Open from the Validator Connection Establishment Engine to the Run Time Validator shall be provided to indicate whether or not a particular consumer of the produced data has an active open connection.
FRS323	If the producer is multi-cast, Consumer_Open shall exist for each multi-cast consumer, 1 through Max_Consumer_Number (indexed from 0 to Max_Consumer_Number – 1).
FRS324	The Consumer_Open indication shall act as an enable to the Run Time Validator on a connection by connection basis. After successful connection establishment, Consumer_Open flags shall transition from Closed to Open.
FRS326	The Validator Connection Establishment engine shall provide an array of consumer connection status resources for multi cast producers that shall be allocated when a consumer makes a connection



Requirement Number	Requirement
FRS327	The consumer numbers (refer to A.4.1) allocated to a Run-time validator shall be the number sent to the consumer in the connection establishment reply.

TST87 The designer shall confirm that the connection establishment engine communicates connection status properly.

#### **F-4.3.7 TST127 – Extended Format Connection Establishment**

Requirement Number	Requirement
FRS361	Safety Originators that connect with an EtherNet/IP target or a DeviceNet target with an Ethernet/IP hop within the path and have an RPI of greater than 100 ms shall use the Extended Format.

TST127 Designers of safety originators shall assure that a strategy is in place such that if an RPI greater than 100 ms is selected and the connection runs over EtherNet/IP that the Extended Format will be used.

#### **F-4.4 TST88 - Safety Supervisor Behavior**

Requirement Number	Requirement
SRS66	The state behavior of the Safety Supervisor object shall control the state of entire device All objects in a safety device are required to be subservient to the states and commands of the Safety Supervisor to manage its functions and behaviors.
SRS142	All CIP safety devices shall replace the Identity object state behavior with the behavior defined in the Safety Supervisor object
SRS187	For safety devices which use the SNCT interface, the states of the Safety Supervisor shall be what determine the Module LED status

TST88 The designer shall confirm by inspection or design review that the device behavior is controlled by the safety supervisor.

#### **F-4.5 TST 106 - Safety Supervisor Reset Password**

Requirement Number	Requirement
SRS23	The SNCT interface shall provide a Reset Password service which takes a vendor specific field to reset the password
FRS346	If a Data Size error occurs in the Safety Supervisor Reset_Password command, either error code 0x15 (too much data) or 0x13 (not enough data) shall be returned.
FRS347	If a Vendor data mismatch occurs when processing a Safety Supervisor Reset_Password command, a Privilege Violation error code (0x0F) shall be returned

TST106 The designer shall confirm by functional test that the device correctly implements the Reset Password service

#### **F-4.6 TST89 - Safety Validator Behavior**

Requirement Number	Requirement
SRS80	A Validator shall only accept connections when the Safety Supervisor is in the state that can accept connections which is device dependent (i.e. Executing only, or Execute/Idle)



TST89 The designer shall confirm by inspection or design review that the validator will not process/accept safety connections while the safety supervisor is in a state which doesn't allow it.

#### F-4.7 TST90 - Corrupted Message Handling

Requirement Number	Requirement
FRS85	If a corrupted message is detected at the standard layer, it shall be discarded and not presented to the application.

TST90 The designer shall confirm that the device will drop corrupted packets and prevent the data from reaching the application.

#### F-4.8 TST91 - Major Fault Effect on Safety State

Requirement Number	Requirement
FRS120	A failure in a background communications diagnostic shall cause all producer/consumer safety connections for that device to be terminated.
FRS121	If there is a major fault in a safety device that causes the device to go into a safety state, the device shall be reset or restarted.
FRS122	In order to clear a major faults in a system the device shall execute and pass all internal self-tests prior to accepting or initiating any safety connections.

TST91 The designer shall confirm that the device drops all safety I/O connections when a diagnostic failure is detected, and doesn't allow them to restart without a reset.

#### F-4.9 TST92 - Configuration Integrity Test

Requirement Number	Requirement
SRS37	For Type 1 configured or tool configured devices, configuration validation of the downloaded safety-relevant data from the software (or originator) shall be performed with SIL3 integrity
SRS40	SIL3 devices shall store/restore its configuration data with SIL3 integrity
SRS41	SIL3 devices shall handle all device state transitions with SIL3 integrity
SRS90	The SCCRC shall be calculated over the configuration data by the device whenever a device configuration is validated (i.e. Validation Request to the safety supervisor or type 1 safety connection)
SRS91	The SCCRC shall be saved to NV storage along with the other NV attributes whenever a configuration is applied (i.e. apply request to the safety supervisor).
SRS124	The reading of the UNID attribute in the Safety Supervisor shall be done with safety integrity.
SRS125	The applying of configuration data and UNID to NVS shall be done with safety integrity.

TST92 The designer shall confirm via inspection or design review that the configuration validation and SCID calculations are done with SIL3 integrity



**F-4.9.1 TST47 – CPCRC Handling in Originators**

Requirement Number	Requirement
SRS93	The Connection Parameter CRC attribute shall be calculated by this originator device whenever the originator's configuration is validated (i.e. Validation request to the safety supervisor)
SRS94	The CPCRC shall be saved to NV storage along with the other NV attributes whenever a configuration is applied (i.e. apply request to the safety supervisor)

TST47 The CPCRC Handling Test in Originators shall confirm that the originator calculates and stores the CPCRC properly when the configuration changes.

**F-4.10 TST93 - Safety Device Hardware Validation Tests**

Requirement Number	Requirement
FRS244	The clocks in safety devices shall be accurate to within 0.02% (0.0002)
FRS248	Each product that produces or consumes safety data shall derive a periodic timer that increments a counter (Producer_Clk_Count or Consumer_Clk_Count) every 128 uSecs.
FRS249	Producer_Clk_Count (and Consumer_Clk_Count) shall be 16 bits in length

TST93 The designer shall confirm via inspection or design review that the clock has a 128 uS tick rate into a 16-bit register with an accuracy within the required 0.05%

TST94 The product design/specification shall insure that it implements both a Module Status and Network Status LED.

**F-4.10.1 TST103 - Switch Behavior Tests**

Requirement Number	Requirement
SRS111	When the switches are read, if the specified MAC ID differs from the value stored in the DeviceNet object's MAC ID Attribute (NVRAM values), the appropriate attribute/s shall be updated and saved to NVRAM

TST103 The designer shall confirm that if MacId switches are used, the required behavior is implemented

**F-4.11 TST95 - Configuration Software Tests**

Requirement Number	Requirement
FRS111	The SNCT which generates a safety device configuration shall generate a unique Safety Configuration Identifier
FRS161	The SCTS shall be a Time/Date stamp marking the Time & Date of the configuration creation or change.
FRS162	The configuration software shall follow the CIP defined Date and Time format (IEC 1131-3) for setting the signature.
SRS1	The configuration Software for a Safety Connection Originator shall provide a means for the following parameters to either be configured or given default values: Expected Packet Interval, Target UNID, Originator UNID, Timeout_Multiplier, Ping Interval EPI Multiplier, Time Coord Msg Min Multiplier, Max Consumer Number, Network Time Expectation Multiplier, ASYNC, Max_Fault_Number (Extended Format only)
SRS21	Software shall transparently insert "password" in all services which require one until a user sets it to something different



Requirement Number	Requirement
SRS39	The SNCT software shall provide a facility to allow the user to confirm downloaded configurations are correct.
SRS47	The SCID shall change when a device's configuration data changes
SRS49	If a SYNC parameter is not defined in a Device's EDS file, the software shall use a default value of 1.
SRS71	SNCT shall always use all 0xFF for the OUNID

TST95 The Software Designer shall confirm that certain key requirements are being met by the software.

1. Confirm the Time&Date Timestamps are generated when a new configuration is created or changed.
2. Confirm the Time&Date value generated uses the IEC 1131-3 format.

#### **F-4.11.1 TST46 - Incorrect Parameter Tests for Originator Connection Configuration**

Requirement Number	Requirement
FRS230	The legal range of the base format Timeout_Multiplier values shall be 1, 2, 3, or 4. The legal range of the Extended Format Timeout_Multiplier value shall be 1 to 255 so long as the Network Time Expectation value does not exceed 5.8 seconds.
FRS231	For any producer, each producer connection and the corresponding consumer connection(s) shall be of the same Connection Type.
FRS235	A legal Ping_Interval_EPI_Multiplier shall have a minimum value determined from the equation $\text{Max\_Consumer\_Number} * \text{Timeout\_Multiplier.PI} + 15$ , and a maximum less than 1000. The default values shall be 100 for Multi-cast and 19 for Single Cast
FRS239	The Ping_Interval_EPI_Multiplier shall be greater than or equal to: $[(\text{the Max Timeout\_Multiplier.PI}(C\#) \text{ for all consumers}) * \text{Max\_Consumer\_Num}] + 15$
FRS241	If the connection type is single-cast, the Max_Consumer_Number shall be equal to 1.
FRS242	For multi-cast connections, the legal range of Max_Consumer_Num values shall be from 1 through 15.
FRS243	The default for the Time_Coord_Msg_Min_Multiplier shall be 0, and the legal range of values shall be from 0 through 7813, which equates to 0 through 1 Sec
FRS318	The Ping_Interval_EPI_Multiplier shall be set small enough to ensure that the Ping_Count_Interval is less than or equal to 100 seconds.
SRS85	Devices implementing the safety extensions to the CCO object shall perform a range check on the Ping Interval EPI Multiplier during a Validation request
SRS86	Devices implementing the safety extensions to the CCO object shall perform a range check on the Time Coord Msg_Min Multiplier during a Validation request.
SRS87	Device implementing the safety extensions to the CCO object shall perform a range check on the Network_Time Expectation_Multiplier during a validation request.
SRS88	Device implementing the safety extensions to the CCO object shall perform a range check on the Timeout_Multiplier during a validation request.
SRS89	Device implementing the safety extensions to the CCO object shall perform a range check on the Max Consumer Number during a validation request.

TST46 The Incorrect Parameter test shall set the SafetyOpen parameters to invalid or out-of-range values and confirm proper detection in the originator at configuration time.



**The following sequence should be performed:**

1. Request Safety Reset type 1.  
**Pass:** Success\_Response.
2. Request Propose/Apply valid TUNID.  
**Pass:** Success\_Response.
3. Request Configure\_Request.  
**Pass:** Success\_Response
4. Request Create service with valid data  
**Pass:** Success\_Response, instance ID.
5. Request Set\_Attributes\_All service, specifying a Timeout Multiplier = 8 (value 1).

**Note: For each of the following Set\_Attributes\_All requests, expected responses are:**

- Pass:** Success\_Response, or Error\_Response, 09, Invalid Attribute Value.
6. If Success\_Response, Request Validate\_Configuration.  
**Pass:** Error\_Response, D0 02
  7. Request Set\_Attributes\_All service, with Transport Class = 1, 2, 3, 4, 5, 6, 7.
  8. Request Set\_Attributes\_All service, with Transport Trigger = 0, 1, 3, 4, 5, 6, 7.
  9. Request Set\_Attributes\_All service, with Ping\_Interval\_Multiplier = invalid min value.
  10. Request Set\_Attributes\_All service, with Ping\_Interval\_Multiplier = 1001 (invalid max value).
  11. Request Set\_Attributes\_All service, with Ping\_Interval\_Multiplier = 1000 and data RPI = 200ms.
  12. Request Set\_Attributes\_All service, with TimeOut Multiplier = 4 for base format connections and Timeout\_Multiplier = 255 for Extended Format connections, MaxConsumers =15, Ping\_Interval\_EPI\_Multiplier = 70.
  13. Request Set\_Attributes\_All service, with a Single Cast connection and MaxConsumers = 3.
  14. Request Set\_Attributes\_All service, with a Multi Cast connection and MaxConsumers = 17.
  15. Request Set\_Attributes\_All service, with Time Coord Min Message Multiplier = 8000.
  16. Request Set\_Attributes\_All service, with Time Coord Min Message Multiplier = 0, MaxConsumers = 15, data RPI = 2ms.



#### F-4.12 TST96 - EDS File Requirements

Requirement Number	Requirement
SRS206	The EDS file for safety devices shall include a [Device Classification] section which shall contain a <b>Classx keyword that indicates the Safety classification.</b>
SRS207	An "EDSFileCRC=" keyword and value shall be included in all safety device EDS files.
SRS208	A "SafetyCfgAssembly=" keyword entry shall be included for Safety devices that are configured via EDS file. This value shall also match an AssemN entry keyword specified in the [Assembly] section.
FRS349	Safety devices that are configured via EDS file shall define a Configuration Assembly

TST96 The EDS Checker shall confirm the EDS file contain all the required entries for safety devices.

#### F-4.13 TST97 - Safety Manual Inspection Requirement

Requirement Number	Requirement
FRS112	The user safety manual shall contain instructions instructing the user "The replacement of safety devices requires that the replacement device be configured properly and operation of the replacement device shall be user verified
FRS103	The safety manual shall contain user instructions that state "If you choose to configure safety connections with an SCID=0, you are responsible for ensuring that originators and targets have the correct configurations", or similar equivalent wording.
FRS154	The user safety manual shall contain an advisory that states "The user should assign SNN numbers for each safety network or safety sub-net that are unique system-wide", or words to that effect.
SRS38	When a SIL3 device is configured directly from a workstation, the device safety manual shall instruct the user to compare the transferred SCID and configuration data with the SCID and configuration data originally viewed in the workstation.
SRS42	The device Safety Manual shall contain an advisory that user testing is the means by which all downloads are validated
SRS43	The device Safety Manual shall contain an advisory that the signature should only be considered "verified" (and configuration locked) after user testing
SRS44	The device User manual shall contain an advisory that configuring an originator with connection data and/or target configuration data must be downloaded to the target so it can be tested and verified. Only then can SCIDs from the target be confirmed.
SRS50	The safety manual shall contain a user instruction requiring the user to completely test a device's operation before setting the Lock Attribute
SRS51	The safety manual shall contain a user instruction requiring the user to upload and compare the configuration from each affected safety devices to that which was sent by the SNCT before setting the Lock Attribute in those devices.
SRS52	The safety manual shall contain a user instruction requiring the user to clear any pre-existing configuration from any safety device before installing it onto a safety network.
SRS53	The safety manual shall contain a user instruction requiring the user to commission all safety devices with MacId (and Baud Rate if necessary) prior to installing it onto a safety network
SRS92	The user shall be instructed in the safety manual to test safety connection configurations after they are applied in an originator to confirm the target connection is operating as intended.



<b>Requirement Number</b>	<b>Requirement</b>
SRS105	The User Safety Manual shall provide a warning that states “LEDs are NOT reliable indicators and cannot be guaranteed to provide accurate information. They should ONLY be used for general diagnostics during commissioning or troubleshooting. Do not attempt to use LEDs as operational indicators”.
SRS193	The Safety manual shall contain a user warning advising that originators that have an “automatic” SNN setting feature should only use that feature when the safety system is not being relied upon.
SRS202	Devices that can be configured via the SNCT interface shall include a safety manual advisory that instructs the user to lock the device after verification has been completed.
SRS203	Devices that can be configured by a Type 1 SafetyOpen shall include a safety manual advisory that instructs the user to verify that all originator-configured safety devices have their ownership assignments as part of the final verification process.
SRS204	All CIP safety devices shall include a safety manual advisory that instructs the user to visually verify that all configuration data was downloaded correctly.

TST97 The Safety Manual review shall confirm that it contains the required safety statements.

#### **F-4.14 TST98 - Requirements Not Tested at this time**

The following requirements do not have tests defined.

<b>Requirement Number</b>	<b>Requirement</b>
SRS186	CIP Safety Device shall provide a separate Module Status LED and a Network Status LED. In this case, these indicators provide important additional safety-related information to the user that cannot be accomplished with a combined Net/Mod Status indicator.

TST98 This is a placeholder for requirements that shall not be tested

#### **F-4.15 TST12 - Origination of Off-link Connection Test**

<b>Requirement Number</b>	<b>Requirement</b>
FRS169	DeviceNet originators shall use the connection manager Forward_Open service (with “Safety” Network Segment extension) when the target is not on the local network segment.
FRS172	On DeviceNet, the originator shall establish an explicit messaging connection with the router or end-node to deliver the Forward_Open or SafetyOpen service requests.

TST12 The Off-link Connection test shall simulate an off-link target to test a DeviceNet client’s use of the Forward\_Open via and explicit connection to the bridge.

**The follow procedure will be used:**

1. Set up the tester to reside on an off-link subnet with an appropriate bridge or router linking the subnets
2. Configure the originator DUT with a device configuration file and representative connection parameter set
3. Trigger the DUT to make connections
4. Confirm the DUT first opens an Explicit Message connection
5. Confirm a standard Forward Open with a Safety Segment is sent over the explicit connection



#### **F-4.15.1 Test Numbers Removed**

The following tests numbers do not exist in this document. Some were removed during the initial development of the Test Guide, others were consolidated into other tests. This section documents the test numbers that have no associated Tests defined. Test numbers: 11, 12, 35, 46, 47, 49, 59



## F-5 Safety Test Matrix

This section contains a matrix that cross references the tests defined in the Safety Test Guide to the behaviors that may be implemented in a safety device. Use the behaviors at the top of the matrix to determine which tests must be applied to the safety device. If there is a discrepancy between the Safety Test Matrix and the specification in the main sections of this volume then the specification shall take precedence.

### F-5.1 Safety Test Matrix

refer to footnotes at bottom of table							Single cast Producing Target	Multi cast Producing Target	Single cast Consuming Target	Multi cast Consuming Originator	Single Cast Consuming Originator	Single cast Producing Originator	Bridge Interface
Test Names	Test Number	DUT Type	Base Format Test	Extended Format Test	Dnet Device	Enet Device							
Standard DeviceNet Conformance	1				X								
Standard EtherNet/IP Conformance	107					X							
Type 2 Connection Est. Pos. Test	2	Target	X		X	X	ROPFS	ROPFS	ROPFS				
Extended Format Type 2 Connection Est. Pos. Test	113	Target		X	X	X	ROPFS	ROPFS	ROPFS				
Connection Initialization Test	3	Target	X	X	X	X	ROPFS	ROPFS	ROPFS				
Connection Parameters CRC Negative Test	4	Target	X	X	X	X	ROPFS	ROPFS	ROPFS				
Type 2 SCID Check	5	Target	X	X	X	X	ROPFS	ROPFS	ROPFS				
Electronic Key Mismatch	6	Target	X	X	X	X	ROPFS	ROPFS	ROPFS				
Target Connection ID Allocation	7	Target	X	X	X		ROPFS	ROPFS	ROPFS				
Multi-cast Producer, Consumer Number Allocation	8	Target	X	X	X	X		ROPFS					
Producer Multi-Cast Time Correction	99	Target	X		X			X					
DeviceNet Extended Format Multi- cast producer formats, seeding & connections	120	Target		X	X			X					



**Volume 5: CIP Safety, Appendix F: Safety Test Plan**

Type 2 Single-Cast Connection Generation Test on DeviceNet	9	Originator	X	X	X						ROPFS	ROPFS	
Type 2 Single-Cast Connection Generation Test on EtherNet/IP	118	Originator	X	X		X					ROPFS	ROPFS	
Type 2 Multi-cast Connection Generation on DeviceNet	10	Originator	X		X					ROPFS			
Type 2 Multi-cast Connection Generation on EtherNet/IP	119	Originator	X	X		X				ROPFS			
Type 2 Extended Format Single-cast Connection Generation	114	Originator		X	X	X					RO	RO	
Type 2 Extended Format Multi-cast Connection Generation	115	Originator		X	X	X				RO			
Electronic Key Generation Test	100	Originator	X	X	X	X				ROPFS	ROPFS	ROPFS	
SafetyClose Processing by Targets	101	Target	X	X	X	X	ROPFS	ROPFS	ROPFS				
Connection Enable/SafetyClose	105	Originator	N/A	N/A	X	X				RO, C6	RO, C6	RO, C6	
Producer CRC & PID/CID Test	13	Producer	X	X	X	X	ROPFS	ROPFS				ROPFS	
Orig Producer Packet Generation - 1 to 2 Bytes Data	14	Producer	X		X	X	X	X				X	
EF Producer Packet Generation - 1 to 2 Bytes Data	109	Producer		X	X	X	X	X				X	
Orig Producer Packet Generation 3 to 250 Bytes Data	15	Producer	X		X	X	C4	C4				C4	
EF Producer Packet Generation 3 to 250 Bytes Data	110	Producer		X	X	X	C4	C4				C4	
Producer Packet Generation Mode Byte	16	Producer	X	X	X	X	ROPFS	ROPFS				ROPFS	
Producer Packet Time Stamp CRC	17	Producer	X		X	X	X	X				X	
Producer Time Correction CRC	18	Producer	X	X	X	X		ROPFS					
Producer Time Correction Mcast Byte & Mcast Byte 2	19	Producer	X		X	X		X					



**Volume 5: CIP Safety, Appendix F: Safety Test Plan**

EF Producer Time Correction Mcast Byte	111	Producer		X	X	X		X					
Base Format Producer Single-Cast	20	Producer	X		X	X	X					X	
Extended Format Producer Single-Cast	121	Producer		X	X	X	X					X	
Producer Run/Idle Usage	21	Producer	X	X	X	X	ROPFS	ROPFS				ROPFS	
Producer Ping Count Usage	22	Producer	X	X	X	X	ROPFS	ROPFS				ROPFS	
Base format Multi-Cast Production to a Single Consumer	23	Producer	X		X	X		X					
Extended Format Multi-Cast Production to a Single Consumer	122	Producer		X	X	X		X					
Multi-Cast Production to Multiple Consumers	24	Producer	X	X	X	X		ROPFS					
Single-cast Consumer Time Coordination Message Generation	25	Consumer	X	X	X	X			ROPFS		ROPFS		
Consumer Positive CRC/PID/CID Test	26	Consumer	X		X	X			ROPFS	ROPFS	ROPFS		
Multi-cast Consumer Time Coordination Message Generation Test	27	Consumer	X		X	X				ROPFS			
Base format Consumer Incorrect Sequence/Insertion Detection	28	Consumer	X		X	X			X	X	X		
Extended Format Consumer Incorrect Sequence/Insertion Detection	112	Consumer		X	X	X			X	X	X		
Consumer Message Corruption Detection	29	Consumer	X	X	X	X			ROPFS	TBD ROPFS	TBD ROPFS		
Consumer Message Delay Detection	30	Consumer	X	X	X	X			ROPFS	ROPFS	ROPFS		
Producer Time Coordination Response Failure Test	31	Producer	X		X	X	X	X				TBD	
Extended Format Producer Time Coordination Response Failure Test	116	Producer		X	X	X	X	X				TBD	
Producer Time Coordination No-response Test	32	Producer	X	X	X	X	ROPFS	ROPFS				TBD ROPFS	



**Volume 5: CIP Safety, Appendix F: Safety Test Plan**

Base format Single-Cast Consumer Test; >= 3 Bytes, Negative Tests	33	Consumer	X		X	X			C3		X		
Extended Format Single-Cast Consumer Test; >= 3 Bytes, Negative Tests	123	Consumer		X	X	X			C3		X		
Base format Single-Cast Consumer; <= 2 Bytes, Negative Tests	34	Consumer	X		X	X			X		X		
Extended Format Single-Cast Consumer; <= 2 Bytes, Negative Tests	124	Consumer		X	X	X			X		X		
Single-Cast Consumer Communication Failure Test	36	Consumer	X		X	X			ROPFS		ROPFS		
Base Format Multi-Cast Consumer Negative Test	37	Consumer	X		X	X				X			
Extended Format Multi-Cast Consumer Negative Test	117	Consumer		X	X	X				X			
Base Format Multi-cast Consumer PID Test	38	Consumer	X		X	X				X			
Extended Format Multi-cast Consumer PID/Rollover detection	125	Consumer		X	X	X				X			
Multi-cast Consumer Communication Failure Response	39	Consumer	X	X	X	X				ROPFS			
Base format Multi-cast Consumer Time Correction Negative Test	40	Consumer	X		X	X				X			
Extended Format Multi-cast Consumer Time Correction Negative Test	126	Consumer		X	X	X				X			
Multi-cast Producer Time Correction Generation Negative Test	41	Producer	X	X	X	X		ROPFS				TBD ROPFS	
Configuration UNID	42	Target	X	X	X	X	C9, ROPFS	C9, ROPFS	C9, ROPFS				
Input Type 1 Connection Establishment Test	43	Target	X	X	X	X	C9, ROPFS	C9, ROPFS	C9, ROPFS				
Output Type 1 Connection Establishment Test	44	Target	X	X	X	X			C1, ROPFS				



**Volume 5: CIP Safety, Appendix F: Safety Test Plan**

Type 1 Configuration from Originators	45	Originator	X	X	X	X					C9, ROPFS	C9, ROPFS	C9, ROPFS
Identity Object	48	Orig/Target	N/A	N/A	X	X	ROPD	ROPD	ROPD	ROPD	ROPD	ROPD	
MacId, SNN, UNID Behavior	50	Orig/Target	N/A	N/A	X		R0, C9 or C10	R0, C9 or C10	R0, C9 or C10	R0, C9 or C10	R0, C9 or C10	R0, C9 or C10	
IP address, SNN, UNID Behavior	108	Orig/Target	N/A	N/A		X	R0, C9 or C10	R0, C9 or C10	R0, C9 or C10	R0, C9 or C10	R0, C9 or C10	R0, C9 or C10	
Reset Switch	51	Orig/Target	N/A	N/A	X	X	C2, R0	C2, R0	C2, R0	C2, R0	C2, R0	C2, R0	
Baseline Supervisor Test	104	Orig/Target	N/A	N/A	X	X	ROPD	ROPD	ROPD	ROPD	ROPD	ROPD	
Propose/Apply TUNID & Reset Commands	52	Orig/Target	N/A	N/A	X	X	R0, C9 or C10	R0, C9 or C10	R0, C9 or C10	R0, C9 or C10	R0, C9 or C10	R0, C9 or C10	
Configuration Lock/Unlock	53	Orig/Target	N/A	N/A	X	X	R0, C10	R0, C10	R0, C10	R0, C10	R0, C10	R0, C10	
Configure Request	54	Orig/Target	N/A	N/A	X	X	R0, C10	R0, C10	R0, C10	R0, C10	R0, C10	R0, C10	
SNCT Configuration Process	55	Orig/Target	N/A	N/A	X	X	R0, C10	R0, C10	R0, C10	R0, C10	R0, C10	R0, C10	
Setting Passwords	56	Orig/Target	N/A	N/A	X	X	R0, C10	R0, C10	R0, C10	R0, C10	R0, C10	R0, C10	
Mode Change	57	Orig/Target	N/A	N/A	X	X	R0, C10, C8	R0, C10, C8	R0, C10, C8	R0, C10, C8	R0, C10, C8	R0, C10, C8	
Safety Validator Diagnostics	58	Orig/Target	N/A	N/A	X	X	C7, ROPD	C7, ROPD	C7, ROPD	C7, ROPD	C7, ROPD	C7, ROPD	



**Volume 5: CIP Safety, Appendix F: Safety Test Plan**

Dnet Bridge FW-O to S_Open Forwarding	62	Bridge	N/A	N/A	X								X
Dnet Bridge nonDnet-to-Dnet Message Forwarding	63	Bridge	N/A	N/A	X								X
Dnet Bridge Dnet-to-NonDnet Message Forwarding	64	Bridge	N/A	N/A	X								X
Standard CIP Message Rejection	65	Consumer	X	X	X	X			X	X	X		